Personal Computer

PC-8801

NEC

マシン語活用マニュアル

塚越一雄＝著

ナツメ社

Personal Computer
PC-8801
NEC
マシン語活用マニュアル
塚越一雄＝著
ナツメ社

PC-8801
マシン語活用マニュアル

［目次より］

ナツメ社
定価——1400円

ISBN4-8163-0326-X C2054 ¥1400E

PC-8801
マシン語活用マニュアル

発行———1984年3月10日

著者———塚越一雄

発行者———田村正隆

発行所———株式会社ナツメ社

郵便番号101

東京都千代田区神田神保町1-52

電話<03>291-1257

振替　東京3-58661

<落丁・乱丁本はお取り替えします>

定価———**1400円**

[本書を読まれる方へ]

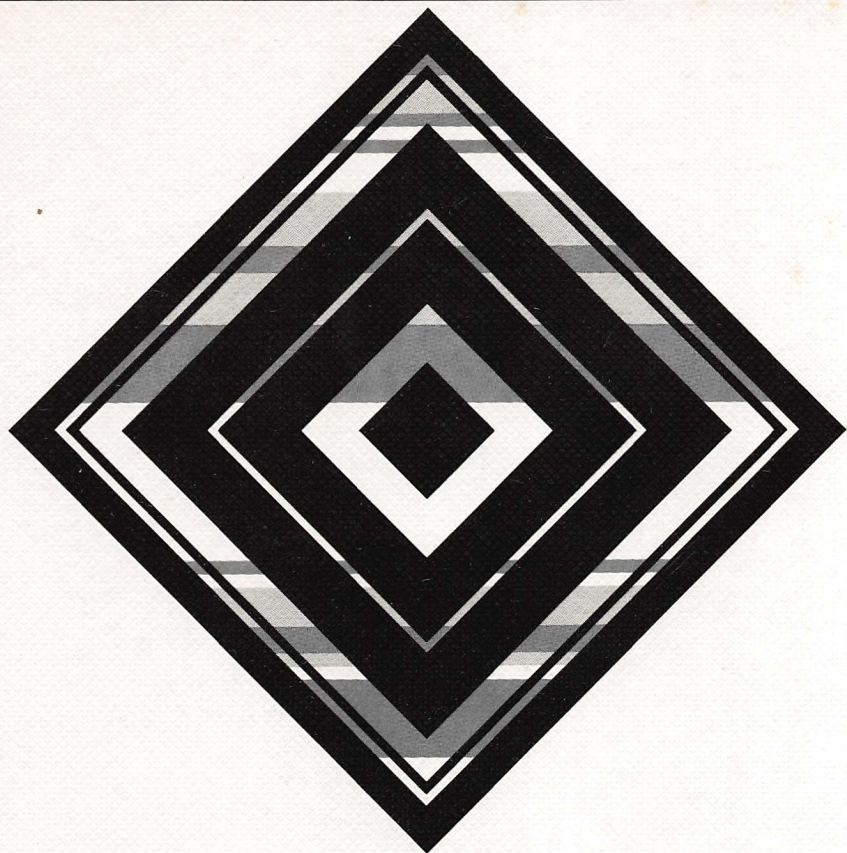
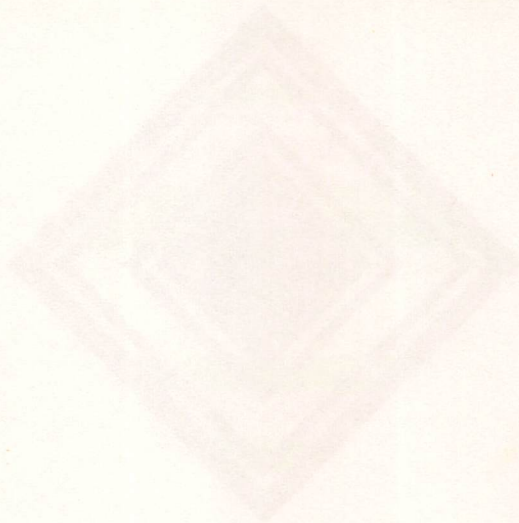
⊙パーソナル・コンピュータの醍醐味、
それは自分のマシンをマシン語で
あやつっているときでしょう。
高級言語の仲介を経ずダイレクトにCPUを制御する、
これこそパソコン・ホビーの原点です。
⊙マシン語の持つ力・魅力については多くのユーザーが
認識し、その必要性を感じるようになってきました。
しかし、その普及はまだまだのようです。
それは、マシン語そのもののむずかしさに加えて、
マシン語が各マシン特有のソフト・ハードの
技術情報を必要とするからです。そして、
これらの情報はなかなか公開されないのが現状です。
⊙本書は、PC-8801を使ってマシン語を
活用したいと願うユーザーに、
マシン語プログラマーへの技術情報を
提供するのが目的です。
⊙そのため本書の前半では、読者とともに、
ROMの解読を試みます。
それは、マシン語プログラマーが
必要とする知識の宝庫です。
さらに、その知識をもとに
各周辺機器を制御する方法を調べます。
⊙こうして本書を活用することで、
PC-8801におけるマシン語の活用法を
修得できるでしょう。

cover design——nobuo nakagaki+yoshifumi hayase

Personal Computer

NEC PC-8801

マシン語活用マニュアル

塚越一雄―著

ナツメ社

MULTIからの手紙

パソコン——パーソナル・コンピュータの醍醐味、それは自分のマシンをマシン語であやつっている時でしょう。高級言語の仲介を経ずしてダイレクトに

CPUを制御する

——これこそパソコン・ホビーの原点です。

それだけではありません。高級言語の死角を補い、ハードウェアの機能を最高に引き出す——これは、マシン語プログラマーだけが甘受できる特権です。

マシン語の持つ力、魅力については多くのユーザーが認識し、その必要性を感じるようになってきました。しかし、その普及はまだまだのようです。それは、

マシン語そのものの難しさ

に加えて、マシン語が各マシン特有の

ソフト・ハードの技術情報

を必要とするからです。そして、これらの情報はなかなか公開されないのが現状です。

本書は、ＰＣ－８８０１を使ってマシン語を活用したいと願うユーザーに、

マシン語プログラマーへの技術情報

を提供するのが目的です。そのため本書の前半では、読者とともに

ROMの解読

を試みます。それは、マシン語プログラマーが必要とする知識の豊庫です。後半では、その知識をもとに各周辺機器を制御する方法を調べます。こうして、本書を活用することでＰＣ－８８０１におけるマシン語の活用法を修得できるでしょう。それが、本書のタイトルである

ＰＣ－８８０１マシン語活用マニュアル

の所以です。

本書の執筆にあたり、ナツメ社の田村正隆氏、また平山正章氏をはじめとする多くのMULTIの仲間にお世話になりました。深謝すると同時に、本書をすべてのＰＣ－８８０１のユーザーに棒げます。

1983年５月

MULTIマイコン研究会事務局代表

塚越一雄

もくじ

第0章

マシン語モニタの解析

これからPC-8801というパーソナルコンピュータを用いて

マシン語プログラム

を作成して行くのに必要な知識をいろいろ探って行くことになります。

PC-8801には、優れた

マシン語モニタ

が内蔵されています。そこには簡易的な機能ながら

アセンブラ

逆アセンブラ

デバッガー

などが内蔵されています。したがって、我々がPC-8801上でマシン語のプログラムを組むには、少なからずこのマシン語モニタのお世話になるわけです。

一方、PC-8801は、アドレス空間の限界から

メモリのバンク切り換え

という手法を使っています。そのため、マシン語モニタの各状態で

現在、どのメモリが選択されてるか？

を知る必要があります。このことについての情報は、マニュアルには書かれていません。また、「メモリのバンク切り換え」の他にも必要な情報（マニュアルには説明されていないもの）がたくさんあります。

これらの情報・知識を獲得するには、やはり自分の力でマシン語モニタを解析する必要があります。

本章では、マシン語プログラマが、PC-8801のマシン語モニタを解析する際の道案内が展開されています。本章を利用し、そして自分の力でマシン語モニタを解析することで、たくさんの必要な情報・知識を獲得するよう努めてください。

0·0 MON処理ルーチンの解析

いよいよ、これからＰＣ－８８０１内蔵の

マシン語モニタのしくみ

を探って行きます。マシン語モニタは、N88-BASICから

MON ⏎

とキーインすることで起動します。すなわち、N88-BASICのコマンドであるMONが、

マシン語モニタの入口

ということになります。したがってN88-BASICインタプリタの

MON処理ルーチン

を調べて行くことで、マシン語モニタ探索の糸口がつかめるのではないか、と予測されます。かくて、我々の第１の目標が設定されました。

「MON処理ルーチンを探る」

これが、これからしばらくの話題となります。

マシン語モニタの起動

```
Disk version [Aug 20,1982]
How many files(0-15)?
NEC N-88 BASIC Version 1.1
Copyright (C) 1981 by Microsoft
45410 Bytes free

Ok
mon

h]■
```

【図0·0】——マシン語モニタの起動

これが、ＰＣ－８８０１のマシン語モニタが起動したところです。ご覧のように、N88-BASICから、

MON ⏎

とキーインすることで

N88-BASICコマンド・レベル →マシン語モニタ・コマンド・レベル

に移ることができます。

MON処理ルーチンへ

ところで、ユーザーが

MON ⏎

とキーインすることは、N88-BASICインタプリタにとっては、どういうことを意味しているのでしょうか？

N88-BASICが、"Ok"を表示し、BASICのコマンドを受けつけることのできる状態にあることを、

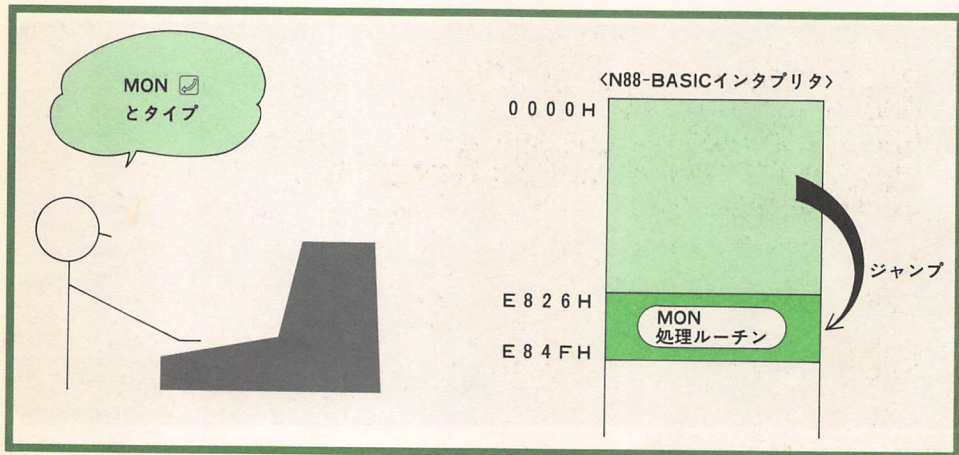
N88-BASICコマンド・レベル

と呼んでいます。この状態でユーザーが任意のコマンドを入力しますと、N88-BASICインタプリタはそのコマンドを解釈し、それぞれの**処理ルーチン**にジャンプして行きます。

〈MONの処理ルーチン〉 →E826H～E84FH

ですから、ユーザーがMONを入力しますと、N88-BASICインタプリタは、

E826Hへジャンプ

することになります。

【図0・1】——MON ⏎の意味

アセンブル・リストを見る

MON処理ルーチンがどんなことをやっているのかは、中身を調べればわかります。それには、その部分のアセンブル・リストがほしいでしょう。アセンブル・リストは、マシン語モニタを使えば見ることができます。

LE826, E84F ⏎

とキーインすれば、アセンブル・リストがＴＶ画面上に表示されます。

また、

P ⏎

とキーインした後に、

LE826,E84F ⏎

とキーインすれば、アセンブル・リストがプリンタに出力されてきます。

MON処理ルーチンの解析

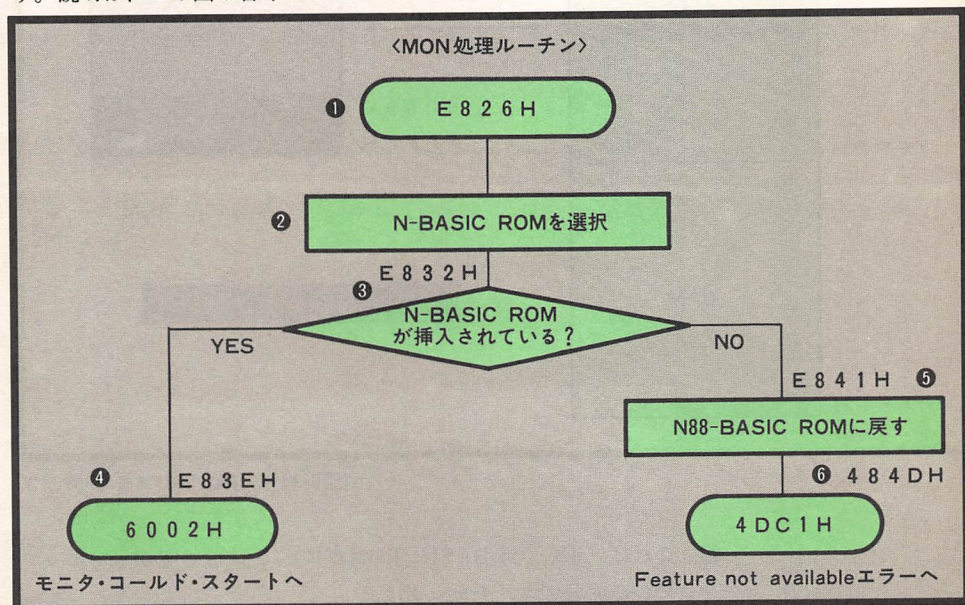
さて、我々の本来の目的は、

マシン語モニタのしくみを探る――――――Ⓐ

ことで、

ＭＯＮ処理ルーチンを調べる――――――Ⓑ

ことではありません。そこで、Ⓑの中からⒶに関係のある部分を中心に「MON処理ルーチン」を解析して行くことにいたします。

まず、「MON処理ルーチン」のフローチャートからお目にかけましょう。説明は、この図の番号にしたがって行います。

【図0・2】――「ＭＯＮ処理ルーチン」フローチャート

❶ MON処理ルーチンのスタート番地です。これをためすには、

USR関数

CALLコマンド

のいずれかを用いればできます。次の図は、CALLコマンドを用い

たものです。N88-BASICインタプリタから、マシン語モニタが起動していることがわかります。

```
I=&HE826:CALL I ←――――――ダイレクトで入力しリターンキーを押す
h]■
```

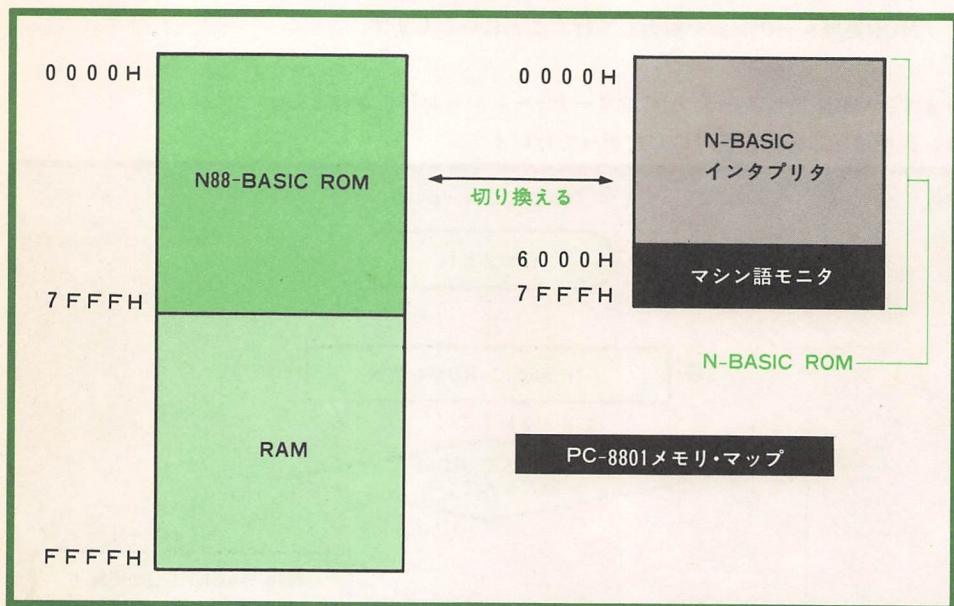
【図0・3】――CALLコマンドを用いて

❷ 「MON処理ルーチン」に入りますと、最初に行うことが、

N-BASIC ROMの選択

です。これは、次の図から明らかなように、マシン語モニタがN－BASIC ROMの

6000H～7FFFH

に入っているからです。

【図0・4】――PC-8801メモリ・マップ

❸ これは、正しくN-BASIC ROMが挿入されているかをチェックしている部分です。チェックは、

```
6000H＝44H
6001H＝42H
```

【図0・5】――N-BASIC ROMの判定

となっているかで判定しています。正しいN-BASIC ROMなら、こ

のように書き込まれているからです。

❹ もし、切り換えたROMがN-BASIC ROMであれば、そこに書かれている**マシン語モニタの処理ルーチン**にジャンプします。それが、6002Hです。すなわち、

マシン語モニタ・コールド・スタート＝6000H

であることがわかります。

❺ もし、切り換えたROMがN-BASIC ROMでなければ、これは6002Hにジャンプさせるわけにはまいりません。そこで元のN88-BASIC ROMに戻します。

❻ そして、4DC1Hにジャンプさせます。

4DC1H＝Feature not availableエラー処理ルーチンのスタート・アドレス

です。すなわち、

利用不可能な機能を指定した

というわけです。CALLを用いて実験してみましょう。ご覧のとおりです。

```
I=&H4DC1:CALL I
Feature not available
Ok
```

【図0・6】――Feature not availableエラー

0·1 BASIC ROM選択の怪

ユーザーが、PC-8801を使う時、N88-BASICと合わせてマシン語を使用すると、より一層その能力を引き出すことができます。マシン語を使うには、マシン語モニタの助けが必要です。しかし、PC-8801のように、

複数のメモリをバンク切換えして使用

している機種では、やはり

マシン語モニタの中身

まで知る必要があるようです。たとえば、

Gコマンド

でマシン語を走らせるにあたっても、現在どのメモリが選択されているのかわからないようでは、うかつにマシン語のプログラムを走らせるわけにまいりません。このようなことがマニュアルに書かれていればよいのですが、現実に書かれていない以上、ユーザーは自衛策として自分でモニタの中身を調べてみる必要があるようです。

これからユーザーがマシン語のプログラムを作る上で必要な部分を中心に

マシン語モニタのしくみ

を調べて行くことにいたしましょう。

N-BASIC ROMの怪

まず次の実験をしてみます。

MON ⏎

でマシン語モニタを起動させ、D(ダンプ・メモリ)コマンドを用いて、

D6000,6001 ⏎

とキーインしてみてください。次のように表示されると思います。

```
MON

hJD6000,6001
6000 03 CD
hJ■
```

【図0·7】――Dコマンドを用いて

これは、6000H～6001Hのメモリの内容を表示させたものです。ごらんのように、

```
6000H=03H
6001H=CDH
```

となっています。これ、おかしいと思いませんか？

前節において我々は、「マシン語モニタはN-BASIC ROMの後部におかれている」こと、そのため「マシン語モニタに入る前にMON処理ルーチンでN-BASIC ROMを選択していること」などを知りました。したがって、モニタが起動している状態では、N-BASIC ROMが選択されているわけです。

またN-BASIC ROMでは、

```
6000H=44H
6001H=42H
```

となっていることも知りました。この値は、図0・7とは明らかに異なります。

いったい、どうなっているのでしょう？

第2の実験

もう1つ実験をしてみます。

まず、次のプログラムを入力してください。

```
hJDBB00,BB0F
BB00 3E 0D DF 3E 0A DF DF 21 BE 79 CD 50 55 C3 1A 4B
hJ■
```

Sコマンドでこの部分を入力する

【図0・8】——実験用プログラム

入力しましたら、

GBB00 ⏎

でプログラムを走らせてみてください。次のように表示されるはずです。

```
hJGBB00

NEC N-88 BASIC Version 1.1
Copyright (C) 1981 by Microsoft
Ok
■
```

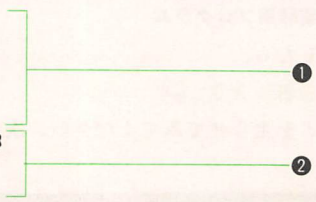
【図0・9】——プログラムを走らせる

N88-BASIC ROMの怪

これは、おなじみのN88-BASICのオープニング・メッセージです。図0・8のプログラムを走らせると、これが表示されるのです。これが表示されるということは、図0・8のプログラムが

N88-BASIC ROM

を利用しているということになります。N88-BASIC ROMの

79BEH～79FBH

には、N88-BASICオープニング・メッセージのデータが入っています。ところが、次に見ますように図0・8のプログラムでは、特にN88-BASIC ROMを選択しているわけではありません。N-BASIC ROMが選択されているマシン語モニタから、いきなり

GBB00 ⏎

で走らせただけです。でも、プログラムはうまく走ったようです。

いったい、どうなっているのでしょう？

実験プログラムの解析

```
                ;-----------------------------------
                ;  N-88 DISK BASIC MESSAGE:1983.5.6
                ;-----------------------------------
                ;
0018            OUTIO: EQU   18H
4B1A            BAS:   EQU   4B1AH            ;BASIC HOT START
5550            MSG:   EQU   5550H
79BE            DN88:  EQU   79BEH            ;79BEH-79FBH
                ;
                       ORG   0BB00H
                ;
BB00 3E0D       EX:    LD    A,0DH    ┐
BB02 DF                RST   OUTIO    │
BB03 3E0A              LD    A,0AH    ├──── ❶
BB05 DF                RST   OUTIO    │
BB06 DF                RST   OUTIO    ┘
BB07 21BE79            LD    HL,DN88  ┐
BB0A CD5055            CALL  MSG      ├──── ❷
BB0D C31A4B            JP    BAS      ┘
                ;
                       END
```

【図0・10】――アセンブル・リスト

これは、図0・8のプログラムのアセンブル・リストです。このプログラム自体は、本節の流れとは関係ありません。しかし、念のため簡単に説明しておきます。説明は、図中の番号にしたがって行うことにいたします。

❶　この部分は、2行改行をおこなっている部分です。ここで用いたシステム・サブルーチンは、次のとおりです。

アドレス　0018H

機　能　**1文字出力**

Aレジスタにキャラクタ・コードを入れてCALLすると、

CRT ――E64CH=0のとき

プリンタ ――E64CH≠0のとき

のいずれかに出力できます。

RST　18H（マシン・コード=DFH）

の1バイトで実行できますから、かなり実用的なサブルーチンといえます。

ちなみに、

キャラクタ・コード=0DHのとき

復帰（carriage return）

カーソルが、行の左端に戻る。

キャラクタ・コード=0AHのとき

改行（line feed）

カーソルを1行下に移す。

となっていることにご注意ください。

❷　この部分は、N88-BASICのオープニング・メッセージを表示しているところです。ここで用いたシステム・サブルーチンは、次のとおりです。

アドレス　5550H

機　能　**メッセージの表示**

このサブルーチンを用いるには、まず表示したいメッセージを任意のメモリ上におきます。メッセージの終りには、

END　MARK=00H

を書き込みます。そして、HLレジスタにメッセージの先頭アドレスをセットしてCALLすれば、メッセージがCRT上に表示されます。

このプログラムでは、

HL=79BEH

にセットしています。実は、**N88-BASIC ROM**の

7 9 B E H～7 9 F B H――――――Ⓐ

には、**N88-BASICのオープニング・メッセージのデータ**が入っています。ここで注意していただきたいのは、Ⓐの部分が、

マシン語モニタのエリア＝6 0 0 0 H～7 F F F H

とスッポリ重なっている、ということです。ですから単純に考えると、このプログラムがうまく走るためには、

N-BASIC ROM ➔N88-BASIC ROM

への切り換えを行う必要があるように見えます。しかし図0・9で実験しましたように、プログラムはうまく動きました。ROMの切り換えを行うことなしに……。

マシン語モニタ解析の必要性

以上、見てきましたようにマシン語のプログラムを走らせるには、

マシン語モニタの中身

を知る必要があるようです。とりわけ**メモリのバンク切り換え**など

マシン語モニタのしくみ

を理解する必要があるようです。さもないと、

●メモリの切り換えを間違えたためにプログラムを暴走させた！

●知らぬ間に異なるROMを解析していて、無駄な時間を費やしてしまった！

という失敗を演じてしまいます。

マシン語モニタを解析する

――これは、PC-8801のようにメモリのバンク切り換えを用いている機種では、マシン語をスムーズに走らせるためのパスポートなのです。

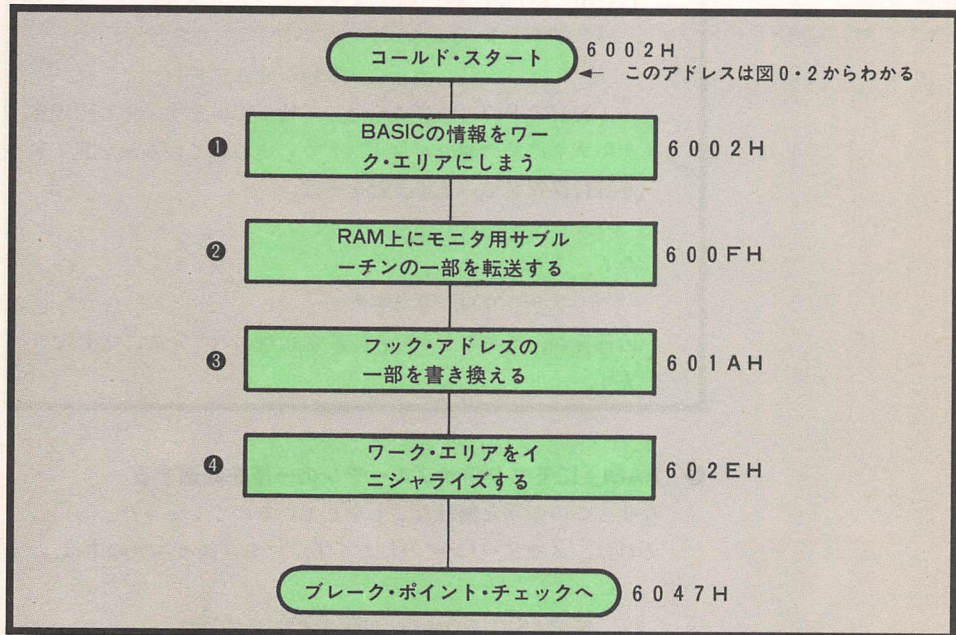
0・2 マシン語モニタの解析

これから具体的に

マシン語モニタの解析

に入って行きます。説明には、フローチャートを用います。そして、フローチャートにつけた番号にしたがって説明して行くことにいたします。

コールド・スタート・ルーチンの解析

まず、マシン語モニタの入口の部分です。フローチャートを次に示します。

【図0・11】――マシン語モニタ導入部

❶ BASICの情報をワーク・エリアにしまう

これは、N88-BASICで使用している次の3つの情報をワーク・エリアにしまっています。コントロールBで

マシン語モニタ →N88-BASIC

に戻った時に、これらの情報を元に戻しています。こうすることで、後にBASICに戻っても、プログラム（N88-BASICインタプリタ）が正しく動きます。

〈ワーク・エリアにしまわれる3つの情報〉

(1) **N88-BASICテキスト・ポインタ：F1F5H**

N88-BASICでは、HLレジスタをテキスト・ポインタに使っています。

(2) **スタック・ポインタ：F1F7H**

具体的にはE5F7Hです。コントロールBでBASICに戻る時、スタック・ポインタの値をこれにセットしています。

(3) **テキスト・ウィンドウの上位アドレス：F1F9H**

ご存知のようにN88-BASICでは、テキストを

0000H〜7FFFH

のRAM上に格納しています。このエリアは、ちょうどN88-BASIC ROMと重なっています。そこでテキストの一部を、

テキスト・ウィンドウ
=8000H〜83FFH

の1Kバイトのエリアに転送して使っています。そして現在、テキストのどの部分がウィンドウに現われているかをF1F9Hに保存しています。たとえば、

F1F9H=20H

なら、テキストの

2000H〜23FFH

の部分が、テキスト・ウィンドウに現われていたことになります。

❷ RAM上にモニタ用サブルーチンの一部を転送する

なぜ、このような無駄なことをしているのでしょう？

それは、メモリ・バンクのためです。マシン語モニタの中は、たえず

N-BASIC ROM ↔ N88-BASIC ROM

の間を行ったり来たりしています。もし、マシン語モニタのすべてのルーチンが、N-BASIC ROM上にあると、元に戻れなくなります。そこで、

メモリ・バンクを行うためのサブルーチン

などを、システム・ワーク・エリアの一部に転送しています。ちなみにバンク切り換えのためのサブルーチンは、次のとおりです。

F16CH：N-BASIC ROMに切り換え
F175H：N88-BASIC ROMに切り換え

❸ フック・アドレスの一部を書き換える

フック・アドレス

ってご存知ですか？　これを自由に使うことで

オリジナルなBASICインタプリタ

オリジナルなマシン語モニタ

が作れます。これについては後述します。とにかくこの部分を概説しますと、次のようになります。大意は、

ＥＤＣ９Ｈ～ＥＤＣＢＨ――――Ⓐ

の３バイトを、N88-BASICとモニタでは異なる用途で使っている、ということです。そのためN88-BASICで使っていたⒶの部分を

Ｆ１ＦＡＨ～Ｆ１ＦＣＨ――――Ⓑ

の部分に移します。しかる後に、

６１３２Ｈ～６１３４Ｈ――――Ⓒ

に入っている３バイト（これがモニタ用です）をⒶの部分に移しておしまいです。

なお、Ⓐの各モードにおける用途は、次のとおりです。

	N88-BASIC		マシン語モニタ	
ＥＤＣ９Ｈ	Ｃ９Ｈ	何もしない	Ｃ３Ｈ	コマンド・エラー・ルーチンへジャンプする
ＥＤＣＡＨ	ＦＦＨ		ＢＣＨ	
ＥＤＣＢＨ	００Ｈ		Ｆ１Ｈ	

❹ ワーク・エリアをイニシャライズする

ワーク・エリアをイニシャライズするとともに、

スタック領域の最上位＝０００１Ｈ

にします。これは、Ｘコマンドで使用される

PC（プログラム・カウンタ）＝００００Ｈ

にするためです。

この部分でイニシャライズされるワーク・エリアは次の通りです。

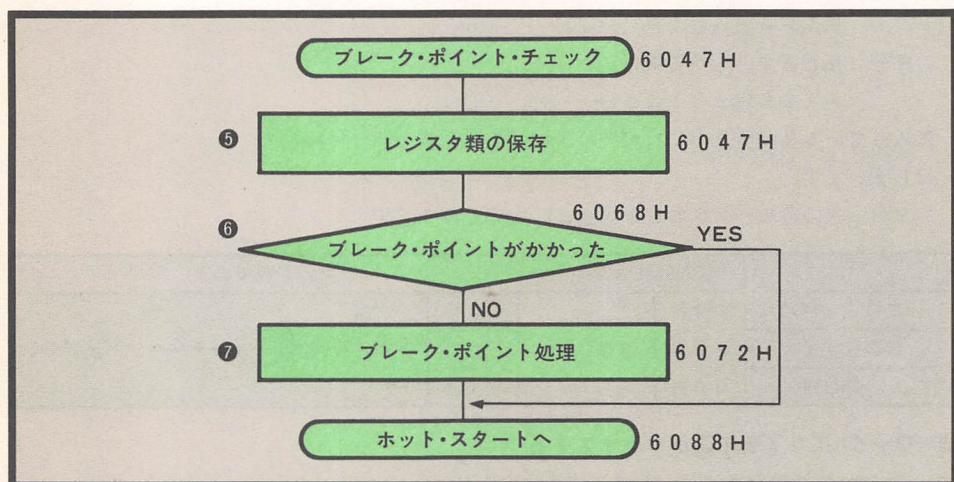
Ｆ１ＣＥＨ	●Ｂ（ベース） ●数値形式のフラグで、ここでは ４８Ｈ＝ｈ〕(16進数) に設定しています。
Ｆ１Ｆ４Ｈ	●Ｐ（プリンタ・スイッチ） ●ここでは、００Ｈに設定していますから、プリント・アウトされません。
Ｆ１Ｅ１Ｈ	●ＲＯＭの選択 ●N88-BASIC ROMに設定しています。

F1DEH	●直前に実行されたコマンドをストア ●これは、Dコマンド、Lコマンド実行直後に ⏎ を押した時、再度同じ内容を実行させるためです。このチェックは、60E0Hのルーチンで行っています。

【図0・12】――イニシャライズされるシステム・ワーク

ブレーク・ポイント・チェック・ルーチンの解析

Gコマンドでブレーク・ポイント・アドレスを設定した場合、またコールド・スタートの後や各コマンドの実行後などにこのルーチンを通過します。

この部分のフローチャートは、次のとおりです。

【図0・13】――ブレーク・ポイント・チェック・ルーチン

❺ レジスタ類の保存

Xコマンドで表示されるレジスタ類の値を

F1FD～F215H

へ保存します。

❻ ブレーク・ポイントの判定

判定は、システム・ワーク・エリアF1CFHの値で行います。

F1CFH	00H――ブレーク・ポイント・アドレスが設定されていない 01H――ブレーク・ポイント・アドレスが1つ設定されている 03H――ブレーク・ポイント・アドレスが2つ設定されている

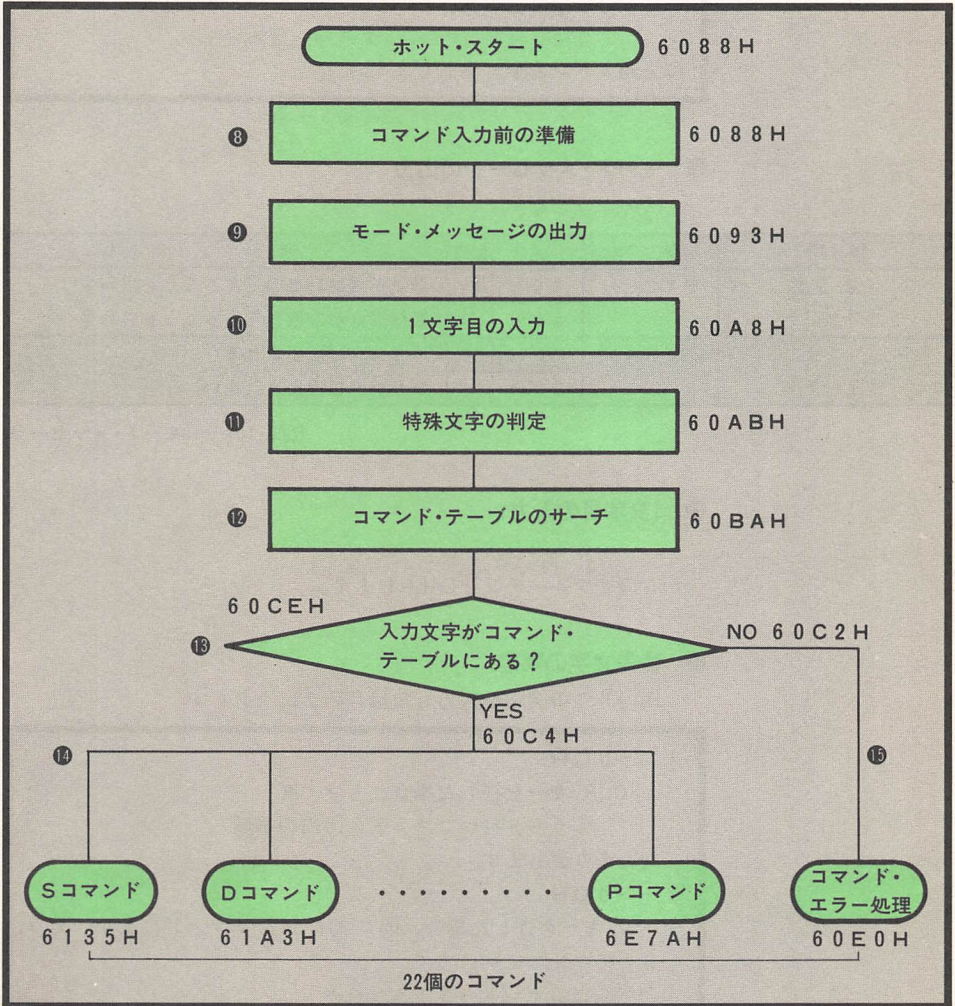
❼ ブレーク・ポイント処理

Gコマンド実行時に、ブレーク・ポイント・アドレスが設定されていた時の処理。ユーザー・プログラムの該当するアドレスに

FFH

が書き込まれていますから、もとのマシン・コードに戻します。

コマンド解析部の解析

まずは、例によってフロー・チャートです。

【図0・14】——コマンド解析部

⑧ コマンド入力前のチェック

ここから、コマンドを入力し、各コマンドの処理ルーチンへジャ

ンプさせるためのコマンド解析部に入ります。マシン語モニタの中心をなす部分です。

ここは、その準備に当たる部分で、次の2つの処理を行います。

●**スタック・ポインタの初期化**
システム・ワーク・エリアＦ１Ｆ７Ｈにしまっておいた値に設定し直します。
●**コマンド・エラーの準備**
スタック領域の先頭に
６０Ｅ０Ｈ＝コマンド・エラー処理ルーチン
の先頭アドレスをセットします。

❾ モード・メッセージの出力

次のようになっています。

種　類	システム・ワーク・エリア	値	出力	意　　味
16進・8進フラグ	Ｆ１ＣＥＨ	４８Ｈ	h	数値形式が16進数であることを表わす
		５１Ｈ	q	数値形式が８進数であることを表わす
プリンタスイッチ	Ｆ１Ｆ４Ｈ	ＦＦＨ	)	プリンタに出力される
		００Ｈ	]	プリンタに出力されない

【図0・15】――モード・メッセージ

❿ 1文字目の入力

６ＦＤＤＨ
からのサブルーチンをCALLします。

⓫ 特殊文字の判定

次の３つの文字を入力した場合のチェックです。

1――０ＣＨ
(CLR)キーを押した場合。もう一度
６０８８Ｈ＝コマンド入力前の準備
からやり直します。
2――０ＤＨ
⏎キーを押した場合。前に実行したコマンドにより、次の３つのケースに別れます。
Ｄコマンド＝６１Ａ９Ｈへ
Ｌコマンド＝６９７ＣＨへ
その他　　＝６０８８Ｈへ
3――２０Ｈ
スペース・キーを押した場合。０ＣＨと同じ。

⓬ コマンド・テーブルのサーチ

各アドレスは、次のとおりです。

コマンド・テーブル：60F0H～6105H――――Ⓐ ジャンプ・テーブル：6106H～6131H――――Ⓑ

⓭ 入力文字がコマンド・テーブルにあるかのチェック

このアルゴリズムは、N-BASICのマシン語モニタで用いているものとまったく同じです。

CPIR（ブロック・サーチ命令）

を用いて、Ⓐのエリアをサーチします。うまくコマンドが見つかると、CPIRのループから脱出してきます。その時、

BCレジスタ＝CPIRのカウンタ

の値が、テーブル・エンドからの距離ですから、それを2倍し、HLレジスタにセットして、Ⓑのジャンプ・テーブルからジャンプ先のアドレスをひろってきます。

⓮ 各処理ルーチンにジャンプする

⓭から、各コマンドの処理ルーチンが、わかります。

コマンド	アドレス	コマンド	アドレス
S	6135H	V	6806H
D	61A3H	W	6754H
X	6254H	R	6807H
F	63D8H	T	727CH
G	6406H	L	6976H
O	6491H	A	6D02H
I	6479H	コントロールD	E672H
E	64D8H	コントロールW	E675H
HELP	746FH	コントロールR	E678H
M	64A6H	コントロールB	6E5FH
B	695EH	P	6E7AH

(注)コマンド・テーブルの順

【図0・16】――モニタ・コマンド・ジャンプ・テーブル

⓯ コマンド・エラー処理

1字目にコマンド・テーブルにないキャラクタを入力すると（⓫の特殊文字を除いて）、

60E0H＝エラー処理ルーチン

へジャンプします。

マシン語モニタの出口

以上までで、マシン語モニタの流れは全部出つくしました。

もう一度、図0・14をご覧ください。⑭のように、各コマンド別にそれぞれの処理ルーチンへジャンプして行きます。各処理が終了しますと、また

⑧＝6088H

に戻ってきます。もっともブレーク・ポイント・アドレスで停止した時のようにスタック・ポインタの値が変化する場合には、図0・13の

⑤＝6047H

に戻って来ますが……。

こうして、マシン語モニタ実行中は、図0・14の

⑧～⑭

の間をグルグルまわっているわけです。

そして、最後。⑭でコントロールBと判定された時、

N88-BASIC インタプリタ

に戻ってきます。つまり、我々が図0・11で解析した

6002H＝コールド・スタート

が、マシン語モニタの入口なら、次に解析する

6E5FH＝コントロールB

がマシン語モニタの出口というわけです。

コントロールBの解析

図0・17がそのフローチャートです。例により、この番号に対応させて見て行きます。

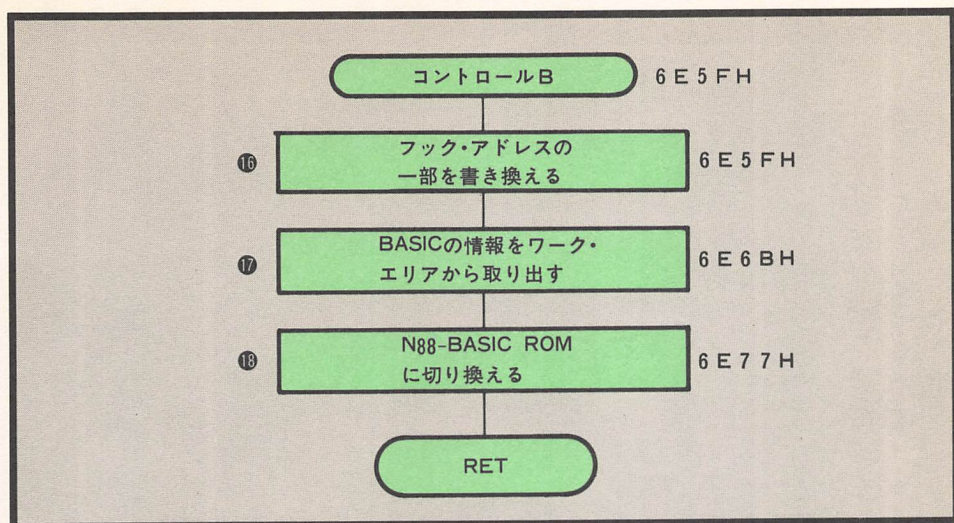
⑯ フック・アドレスの一部を書き換える

これは、図0・11の③でマシン語モニタ用に書き換えたフック・アドレスの一部を元に戻します。具体的には、

〈N88-BASIC用の値〉		〈フック・アドレス〉
F1FAH	――→	EDC9H
F1FBH	――→	EDCAH
F1FCH	――→	EDCBH

のようなことをしています。

⑰ BASICの情報をワーク・エリアから取り出す

これも図0・11の①の作業と逆のことをすることになります。す

【図0・17】――コントロールB

なわち、マシン語モニタの中で壊された次の3つの情報をもとに戻してやります。

(1)N88-BASICテキスト・ポインタ：F1F5H
(2)スタック・ポインタ：F1F7H
(3)テキスト・ウィンドウに、もとのテキストを写影する

```
LD   A,(F1F9H)
OUT  (70H),A
```

⑱ N88-BASIC ROMに切り換える

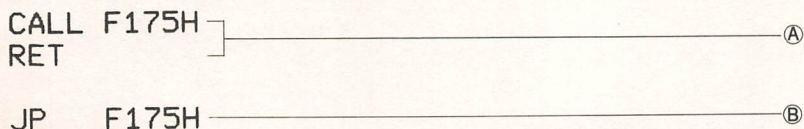
具体的には、

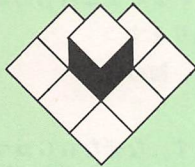
```
CALL F175H
```

です。あとは、すでにスタック・ポインタが復活していますから

```
RET
```

でN88-BASICに戻ることができます。ただし、

```
CALL F175H ┐
RET        ┘―――――――――――――Ⓐ

JP   F175H ――――――――――――――Ⓑ
```

ⒶとⒷは、同値ですからマシン語モニタでは、Ⓑを採用しています。

Gコマンド処理ルーチンとROMセレクトの秘密

前章において我々は、マシン語モニタ活用のために

マシン語モニタのしくみ

を探ってきました。その流れがわかってきたところで、いよいよこれらの知識を応用し行くことになります。

本章では、初めに

Gコマンド

を解析します。なぜならGコマンドこそ、我々の知りたい技術情報の宝庫だからです。そこには、マシン語プログラマにとって必要な技術情報がたくさん隠されています。Gコマンドを解析することで、それらを獲得して行くことができます。

次いでその成果として、マシン語モニタにおけるROMセレクトの秘密をあばくことになります。Gコマンド実行直後のROMの状態――それはこの部分で明らかになります。

本章の最後では、Gコマンドに付随する

ブレーク・ポイントの使い方

を調べます。なぜならブレーク・ポイントはマシン語プログラマーにとって強力な武器であるにもかかわらず、あまりポピュラーではないからです。これをマスターすること――これは次章理解のためのキーポイントとなることでしょう。

1・0 Gコマンドの解析

前章において、我々は

マシン語モニタの流れ

を解析してきました。ここで得られた種々の技術情報を利用して行けば、だんだんと

PC-8801におけるマシン語活用法

の正体が明らかになってきます。

我々は、これから数あるマシン語モニタ・コマンドの中から

Gコマンド

のしくみを解析して行くことになります。実は、ここが重要なポイントとなります。なぜならGコマンドこそ、

メモリ・バンクの秘密

ブレーク・ポイントのしくみ

マシン語プログラムの実行を止める

などの秘密の鍵となっているからです。

Gコマンドを解き明かせるか？

——これが、PC-8801における

マシン語活用の運命の別れ道

になっているのです。

F1C2H～F1C7Hの秘密

Gコマンド解析の前に、まずは何も考えずに、言われるがままに、指示通りに、素直に次のことを実行してください。図1・0の出力画面を参考に。

❶ N88-BASICを立ち上げる（ROMバージョンでもDISKバージョンでも、どちらでも可）

❷ MON ⏎

でマシン語モニタを起動する。起動したら、何のイタズラもせずに直ちに次の❸に進んでください。

❸ LF1C2,F1C7 ⏎

を実行する。これは、

F1C2H～F1C7H

の部分を逆アセンブルしたことになります。

以上でおしまいです。御協力有難うございました。

```
Disk version [Aug 20,1982]
How many files(0-15)?
NEC N-88 BASIC Version 1.1
Copyright (C) 1981 by Microsoft
45410 Bytes free

Ok
mon

h]LF1C2,F1C7
F1C2   CD F175      CALL F175
F1C5   C3 0000      JMP  0000
h]■
```

N88-BASICを立ち上げる

マシン語モニタを起動

ザイログ表記
CALL　F175H
JP　　0000H

【図1・0】――F1C2H～F1C7Hを逆アセンブル

Gコマンド処理ルーチンの解析

そして、これから

Gコマンドの解析

に入ります。図1・2をご覧ください。これが、「Gコマンド処理ルーチン」のフローチャートです。図の番号に対応させて、解説を加えてまいります。

❶ Gコマンドの入口

これは、第0章で解析いたしました。図0・16のとおり

Gコマンドの入口＝6406H

です。

❷ 各種初期設定

Gコマンドの実行に先だち、次の2つの初期設定をおこなっています。

(1) **ジャンプ・テーブルの一部を書き換える。**

```
E669H=C3H ⎫
E66AH=C8H ⎬ JP  F1C8H
E66BH=F1H ⎭
```

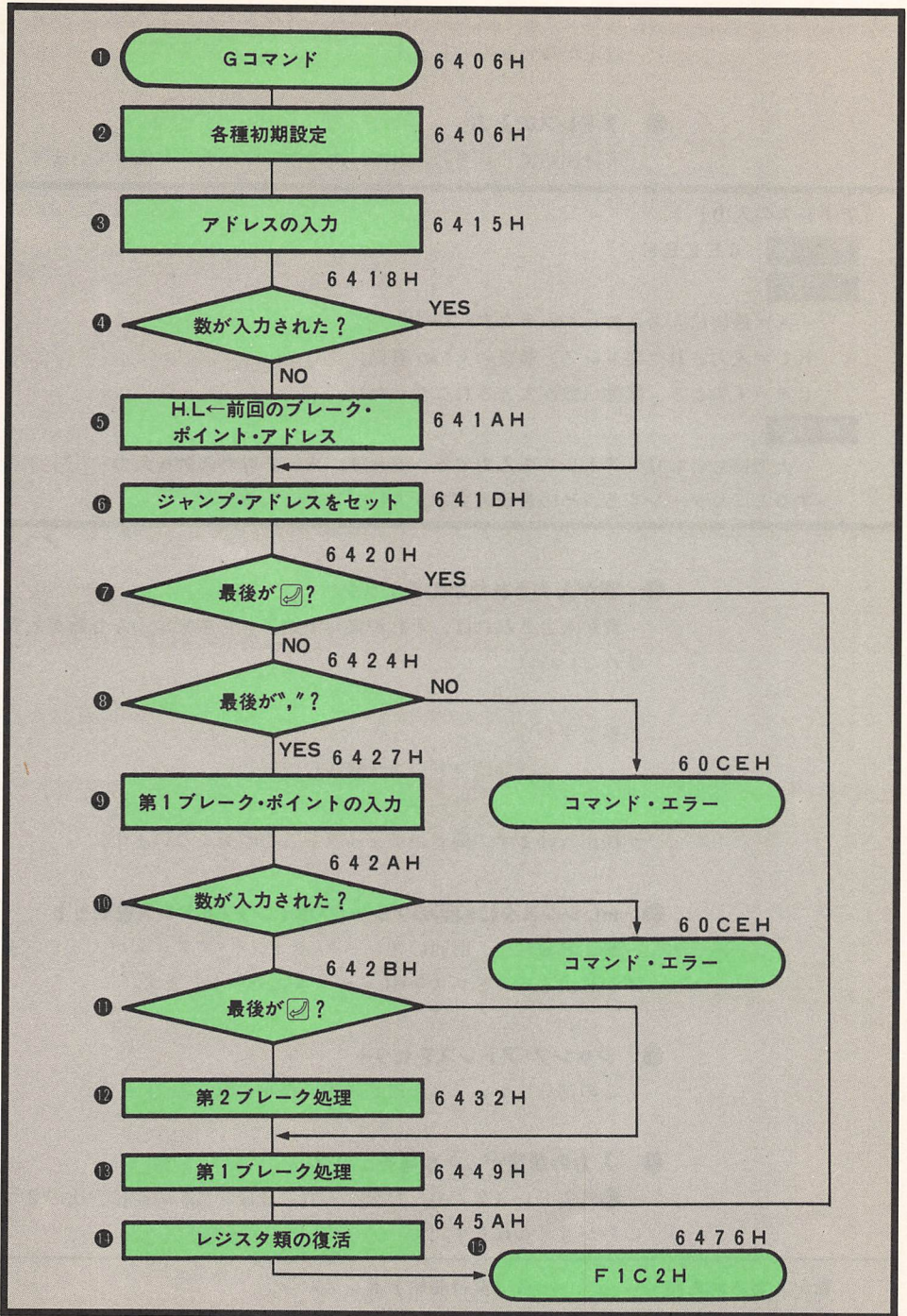
【図1・1】――ジャンプ・テーブル

図1・1のように書き換えます。この部分については、後で

ブレーク・ポイントのしくみ

を解析する時に説明いたします。

(2) **ブレーク・ポイント・フラグのリセット**

❶
Gコマンド
6406H
❷
各種初期設定
6406H
❸
アドレスの入力
6415H
6418H
YES
❹
数が入力された？
NO
❺
H.L←前回のブレーク・
ポイント・アドレス
641AH
❻
ジャンプ・アドレスをセット
641DH
6420H
YES
❼
最後が⏎？
NO
6424H
NO
❽
最後が",″？
YES
6427H
60CEH
❾
第1ブレーク・ポイントの入力
コマンド・エラー
642AH
❿
数が入力された？
60CEH
コマンド・エラー
642BH
⓫
最後が⏎？
⓬
第2ブレーク処理
6432H
⓭
第1ブレーク処理
6449H
645AH
⓮
レジスタ類の復活
6476H
⓯
F1C2H

【図1･2】――Gコマンド処理

Gコマンドの実行にあたり、とりあえずブレーク・ポイントの設定がないものとします。

❸ アドレスの入力

実行開始アドレスの入力で、次のサブルーチンを使っています。

[アドレスの入力]

アドレス 6EEEH

出 力

A＝最後に入力されたキャラクタ・コード。

HL＝入力されたアドレス。最後の4桁が有効。

CY＝1のとき、無効(数が入力されなかった)。

機 能

入力待ちとなり、アドレスを入力する。0〜9、A〜F以外の数を入力すると、リターンする。その最後の文字がAレジスタに入る。

❹ 数が入力されたかのチェック

数が入力されれば、それが実行開始アドレスです。もし数が入力されない時は、

G ⏎

の形ですから、

実行開始アドレス＝最後に実行した
ブレーク・ポイント・アドレス

が採用されます。両者のチェックをここでおこないます。

❺ HLレジスタに前回のブレーク・ポイント・アドレスをセット

F1FFHに、前回のブレーク・ポイント・アドレスが入っていますから、そのアドレスをHLレジスタにセットします。

❻ ジャンプ・アドレスをセット

この部分については、後で説明いたします。

❼ 入力の最後が ⏎ かをチェック

❸でアドレスを入力した際、入力の最後が ⏎ の時は、次の2つの形が考えられます。

数が入力された時――――――G 〈実行開始アドレス〉 ⏎
数が入力されなかった時―――G ⏎

どちらの形も、正しい形ですから⑭に進めます。そうでない場合は、ブレーク・ポイント・アドレスの入力が続くものと考えられますから、次の⑧に進めます。

⑧　入力の最後が、，(コンマ) かチェック

入力の最後が ↵ でなければ

G〈実行開始アドレス〉，〈ブレーク・ポイント・アドレス〉……
　　　　　　　　　　　↑ここが，(コンマ) でないとおかしい

という形が続きますので、入力の最後が，(コンマ) でなければなりません。そのチェックをおこないます。そうでない時は、

コマンド・エラー＝60CEH

にジャンプさせます。

⑨　第1ブレーク・ポイントの入力

，(コンマ) が続いている時は、再び

6EEEH＝アドレスの入力

をCALLして、第1ブレーク・ポイント・アドレスを入力します。

⑩　数が入力されたかのチェック

G〈実行開始アドレス〉，――――
　　　　　　　　　　　↑この部分は、数 (アドレス) でなければならない

【図1・3】――，には、アドレスが続く

このように，(コンマ) の後は、アドレスでなければなりませんから、数が入力されたかのチェックをします。もちろん、数が入力されていない時には、コマンド・エラーとします。

⑪　入力の最後が ↵ かをチェック

⑩で第1ブレーク・ポイント・アドレスを入力した際、入力の最後が ↵ の時は、

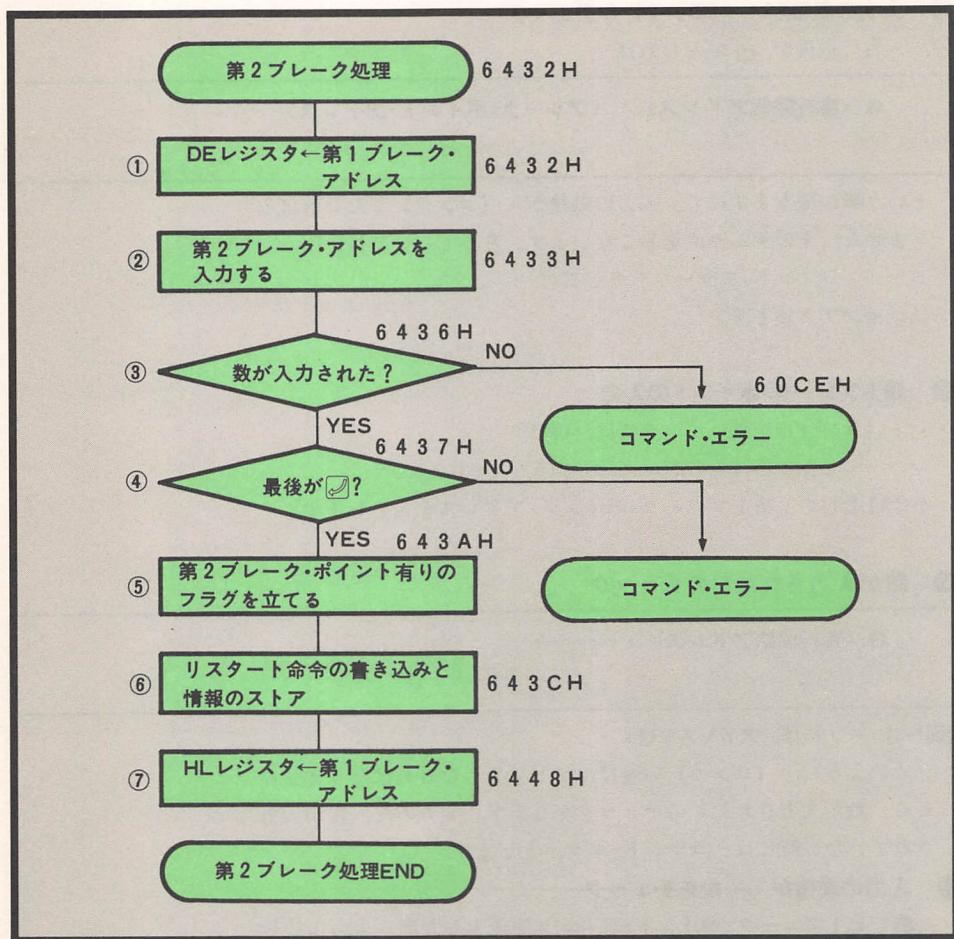
G〈実行開始アドレス〉，〈第1ブレーク・ポイント・アドレス〉 ↵

という形になります。これは、文法的に正しい形ですから後の⑬に進めます。そうでなければ、

G〈実行開始アドレス〉，〈第1ブレーク・ポイント・アドレス〉
　　　　　　　　　　，〈第2ブレーク・ポイント・アドレス〉 ↵

と続くはずですから、次の⑫に進めます。

⑫ 第2ブレーク処理

この部分は、中がいくつかの処理に分かれていますから、より詳細なフロー・チャートを示します。説明は、この番号に分けておこないます。

【図1・4】——第2ブレーク処理

① DEレジスタに第1ブレーク・ポイント・アドレスをストア

第1ブレーク処理は、後で⑬でまとめてやりますので、とりあえず第1ブレーク・アドレス（❾でHLレジスタに入力してある）をDEレジスタに待避させます。

② 第2ブレーク・アドレスの入力

もうおなじみの

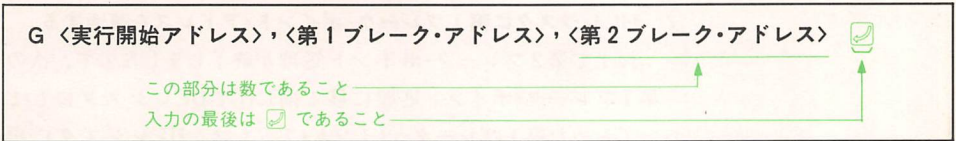
```
CALL 6EEEH
```

でおこなっています。

③ 数が入力されたかのチェック

ここでは、次のような形を想定しています。

G 〈実行開始アドレス〉，〈第1ブレーク・アドレス〉，〈第2ブレーク・アドレス〉 ↵

この部分は数であること
入力の最後は ↵ であること

【図1・5】――フル・オプションの形

ですから、入力されたものが数でなければ

コマンド・エラー＝60CEH

にジャンプさせます。

④ 入力の最後が ↵ かをチェック

同じく図1・5により、入力の最後が ↵ でなければ

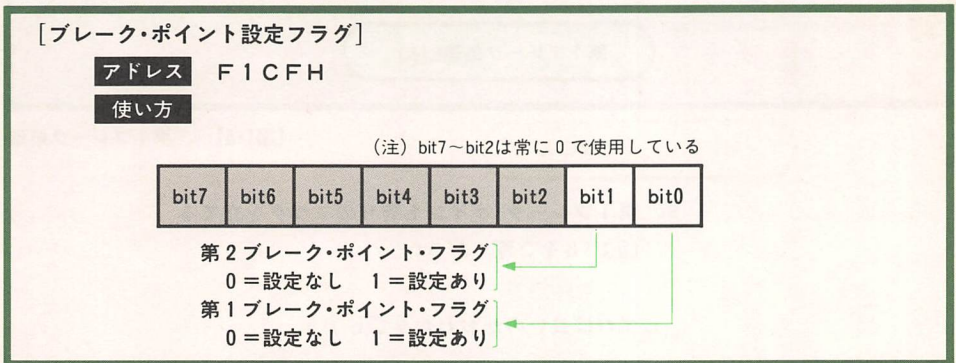
コマンド・エラー＝60CEH

です。

⑤ 第2ブレーク・ポイント有りのフラグを立てる

ブレーク・ポイントが設定されているかどうかは、

システム・ワーク・エリア＝F1CFH

で判定しています。この部分は、先の❷で0にクリアされています。F1CFHの使い方は、次のようになっています。

【図1・6】――ブレーク・ポイント設定フラグ

したがって、ここでは

bit 1 = 1

にセットします。

⑥ **リスタート命令の書き込みと情報のストア**

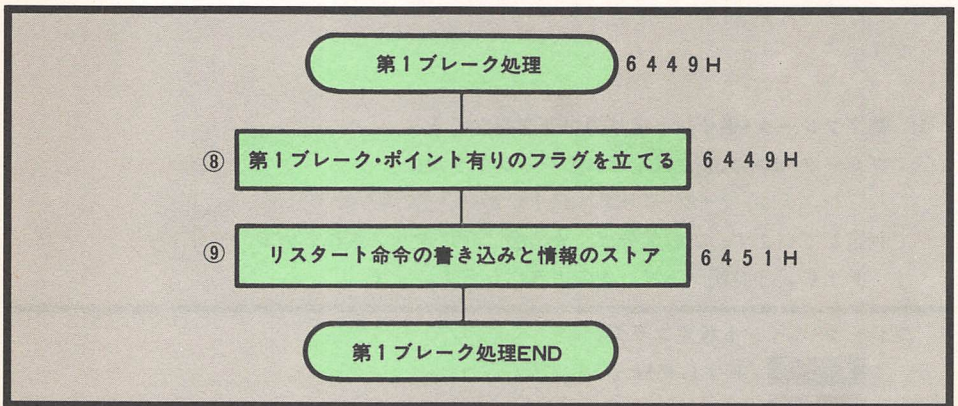
この部分については、後でブレーク・ポイント・アドレスを説明する時に一緒に行なった方がわかりやすいでしょう。

⑦ **HLレジスタに第1ブレーク・ポイント・アドレスを復活する**

以上で第2ブレーク・ポイント処理が終了しましたので、次の第1ブレーク・ポイント処理に移る前に①でDEレジスタにしまっておいた第1ブレーク・ポイント・アドレスをHLレジスタに戻します。

以上が、第2ブレーク処理です。ここから図1・2に戻り、次の⑬に進みます。

⑬ **第1ブレーク処理**

この部分を詳細化したサブルーチンを次に示します。

【図1・5】――第1ブレーク処理

⑧ **第1ブレーク・ポイント有りのフラグを立てる**

図1・6をご覧ください。

bit 0 = 1

にすれば良いのがおわかりでしょう。

⑨ **リスタート命令の書き込みと情報のストア**

ここは、⑥と同じく後で説明いたします。

以上が⑬、第1ブレーク処理です。ここで図1・2に戻り、次の⑭に進みます。

⑭ レジスタ類の復活

前回のブレーク・ポイントで止まった時や、Xコマンドで変更した最終のレジスタ類の値が、

F1FDH～F215H

にストアされています。これらの値を各レジスタに再配置いたします。さもないと、ここまで来る間に、レジスタの値がいろいろ変わってしまっているからです。

⑮ F1C2Hへジャンプ

Gコマンドの1番最後は、

```
JP   F1C2H
```

で、突如としてF1C2Hへジャンプしています。

F1C2H～

は、どんなことがプログラムされているのでしょう？　実は我々は、すでにこの部分の逆アセンブル・リストを見ています。それが、図1・0の逆アセンブル・リストです。再掲いたします。

```
CALL F175H
JP   0000H
```

この部分は、重要ですから良くお読みになってください。まず1行目。このシステム・サブルーチンは、次の働きをします。

［N88-BASIC ROMの選択］

アドレス　F175H

機　能

このサブルーチンをCALLすると、今までのモードに無関係にN88-BASIC ROMのモードに切り変わる。

そして、2行目、これが大変です。何とGコマンドの最後に

0000Hにジャンプ

しているのです。これは、おかしいと思いませんか？　0000Hにジャンプしたら、システムがリセットしてしまい、

N88-BASICがコールド・スタート

します。これでは、せっかくのGコマンドが何年たっても走らないことになります。

1・1　GコマンドとROMセレクトの秘密

いくつかの疑問は残りましたが、前節で一通り

Gコマンド処理ルーチンの解析

が終わりました。そこで、これからそれらの疑問を1つ1つ解決して行くことになります。その過程で自然と

マシン語プログラムのためのキー・ポイント

が浮かび上がってくることでしょう。マシン語モニタ——とりわけ

Gコマンド処理ルーチン

は、マシン語プログラマーのための**技術情報の宝庫**なのです。

F1C2H～F1C7Hの妖怪変化

最初は、図1・0の疑問から解決してまいります。

まず、次のプログラムを入力してみてください。

```
h]DBB00,BB0D
BB00 3E 0D DF 3E 0A DF DF 21 BE 79 CD 50 55 FF
h]■
```

【図1・6】——マシン語プログラム

これは、図0・8のプログラムを少し変更したものです。

GBB00 ⏎

でプログラムを走らせてみてください。次のようにN88-BASICのオープニング・メッセージが表示されます。

```
h]GBB00

NEC N-88 BASIC Version 1.1
Copyright (C) 1981 by Microsoft

h]■
```

【図1・7】——プログラムの実行

図0・9の画面との違い、おわかりですか？

図0・8のプログラム——実行終了後、BASICのコマンド・レベルに
図1・6のプログラム——実行終了後のコマンド・レベルに

の違いがあります。

さて、実はこの実験のねらいは別なところにあります。図1・7の画面が出ましたら、ただちに

LF1C2,F1C7⏎

ときーインしてください。次のように逆アセンブル・リストが表示されます。

```
h]LF1C2,F1C7
F1C2  CD F175      CALL F175
F1C5  C3 BB00      JMP  BB00
h]■
```

図1・0と比較されたし

【図1・8】——逆アセンブル

これと、図1・0と比較してみてください。特に

F1C5H：JMP BB00

の部分を。

もう1つ実験してみます。次のプログラムを入力してみてください。

```
h]DABCD,ABDA
ABCD 3E 0D DF 3E 0A DF DF 21 BE 79 CD 50 55 FF
h]■
```

【図1・9】——マシン語プログラム（その2）

そして、

GABCD⏎

でプログラムを走らせます。

```
h]GABCD

NEC N-88 BASIC Version 1.1
Copyright (C) 1981 by Microsoft

h]■
```

【図1・10】——プログラムの実行

同じ実行結果が得られました。そして、

LF1C2,F1C7⏎

で逆アセンブル・リストを取ります。

```
h]LF1C2,F1C7
F1C2  CD F175       CALL F175
F1C5  C3 ABCD       JMP  ABCD
h]■
```

【図1・11】――逆アセンブル

書き換えられた命令

以上の実験結果をまとめますと、次のようになります。

〈F1C5H～F1C7Hのマシン語〉
MON ⏎ の直後――――JP 0000H［図1・0］
GBB00 ⏎ の直後――――JP BB00H［図1・8］
GABCD ⏎ の直後――――JP ABCDH［図1・11］

これを見ますと、Gコマンドで指定した実行アドレスと同じアドレスにジャンプするように、F1C5H～F1C7Hの部分が変化しているのがわかります。

ここで前節⑮の疑問が解決したようです。⑮では、Gコマンドの処理がすべて終り、最後に

```
CALL  F175H――――N88 BASIC ROMに切り換え
JP    0000H――――0000Hにジャンプ
       └――――この部分が不可思議であった
```

という疑問に突き当たりました。「0000Hにジャンプ」――これはおかしいですね。しかし、今わかりましたようにこの部分は、

実行直前に書き換えられていた！

のです。実行開始アドレスに。命令そのものを書き換える――これができるのも、

F1C5H～F1C7H

の部分がRAM上にあるからです。

実行開始アドレスをセット

では、いつF1C5H～F1C7Hが書き換えられたのでしょう？　それが、前節で保留しておいた⑥の部分です。⑥では、

```
LD  (F1C6H),HL
            └――〈実行開始アドレス〉が入っている
```

ということをやっています。これをマシン・コードに遡って説明いたしますと、次のようになります。

F1C5H	C3	――――JPで固定
F1C6H	(Lレジスタ)	――実行開始アドレスの下位
F1C7H	(Hレジスタ)	――実行開始アドレスの上位

したがって、

JP 〈実行アドレス〉

のようになるのがおわかりでしょう。

バンク切り換えとシステム・ワーク・エリアの機能

以上の我々の解析結果から、Gコマンドはかなり面倒なことをしているのがわかります。

RAM(システム・エリア)上に実行開始アドレスをセット

↓

そこにジャンプすることで実行開始アドレスにジャンプ

という二重手間をかけています。ストレートに実行開始アドレスにジャンプすれば良いのに、なぜこのような一見ムダそうなことをしているのでしょう?

それは、直前に

ROMをN88-BASIC ROMに切り換えている

からです。マシン語モニタは、N-BASIC ROM上にあります。もちろん、Gコマンド処理ルーチンもN-BASIC ROM上にあります。ですから、N88-BASIC ROMに切り換えたとたん、N-BASIC上のモニタ・ルーチンは使えなくなるのです。

そこでN-BASIC ROMの状態にあるうちに、RAM上に必要な情報を書き込んでおき、ROMの切り換え後はRAM上で処理を続行するという手法をとっているのです。

ROM別システム・サブルーチンのCALL

以上のように、PC-8801は、ROMのバンク切り換えという手法を使っているため、マシン語モニタの中身も、非常に巧妙に作られています。システム・ワーク・エリアを巧みに使わないと、複数のROMを制御できないからです。前述の

```
F1C2H:CALL  F175H  ――――Ⓐ
F1C5H:JP    〈実行開始アドレス〉
```

【図1・12】――Gコマンドの最後に飛んでくるルーチン

の部分もその1つです。

ここで図1・12のⒶの部分にご注目を！　大変なことに気がつきませんか？

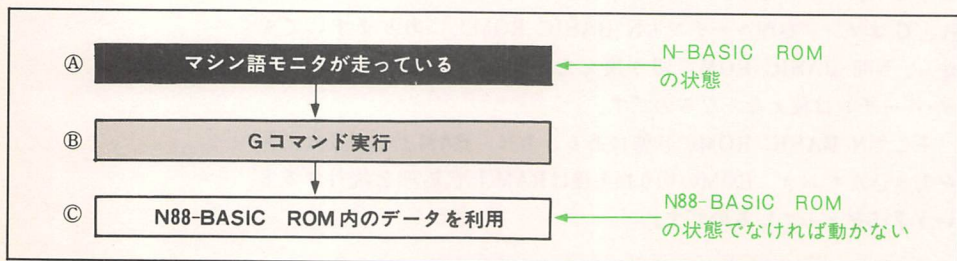
マシン語モニタは、N-BASIC ROMの上で作動しています。しかし、

> ［秘　伝］
>
> **マシン語モニタのGコマンド実行直後は、**
>
> **N88-BASIC ROM**
>
> **がセレクトされている。**

ということに着目してください。これはマシン語を使う上で非常に重要なことで、ROM選択のキーポイントになります。すなわちROM内のシステム・サブルーチンをユーザーがCALLする時は、次のようにしてください。

> **〔N88-BASIC ROM内サブルーチンをCALLする時〕**
>
> **そのままCALLすれば良い。**
>
> **〔N-BASIC ROM内サブルーチン（マシン語モニタ等）をCALLする時〕**
>
> **一度、N-BASIC ROMに切り換えた後にCALLする。**

以上がわかれば、第0章の疑問は解決がつきます。

図0・8、図0・9における実験結果の謎です。もう一度図解してみます。

実験では、これがうまく動きました。しかし、今や我々はこれを十分に説明することができます。すなわち、Ⓐではたしかに ROMは、N-BASICのものがセレクトされていました。しかし、Ⓑを実行したとたん、（我々の解析によると）自動的にN88-BASICのものにROMが切り換わります。したがってⒸが合理的であることは、十分にわかります。

いかがですか？　上記の鉄則

Gコマンド実行直後は、N88-BASICモード

であること、**ゆめゆめ**お忘れなく！

1·2 ブレーク・ポイントの設定

前節では、PC-8801マシン語プログラマーの最も基本となる

Gコマンド実行直後のROMの状態

を探りました。本節では、それに引き続き

プログラム実行停止の方法

を調べて行きます。プログラムの実行停止なんて

HALT（マシン・コード＝76H）

を使えばいいじゃないか、とおっしゃるかもしれません。しかしながら、少しプログラムを組んだことのある人なら、

HALTは、マシン語プログラムの実行を止めるための有力な方法ではない

ことは良くおわかりのことと思います。

これから我々は、

ブレーク・ポイントのしくみ

を探って行きます。実はこの中に、PC-8801における

マシン語プログラムにおける実行停止

の有力な方法が隠されているのです。

X（イグザミン・レジスタ）

さて、その**ブレーク・ポイント**ですが、これが意外と知らない人が多いようです。ブレーク・ポイントは、**Gコマンド**の中で設定します。その説明が、「PC-8801ユーザーズ・マニュアル」のP.15—10に載っています。しかし、この部分、さすがにマシン語プログラマーを対象に書いてあるらしく、きわめてアッサリと書かれています。したがってその使い方が良くわからず、そのためせっかくのGコマンドもプログラムの実行だけに使われているようです。

そこでブレーク・ポイントの説明にあたり、はじめにその使い方を説明しておきましょう。

まずマシン語モニタを起動しましたら、

X ⏎

```
h]X
A :00 F :PZ---E-- B :0000 D :EDCC H :0001 A':46 F':M--H-E-- B':0000 D':0000
H':0000 IX:1340 IY:0000 I :F3 PC:0000 SP:E5F9
h]■
```

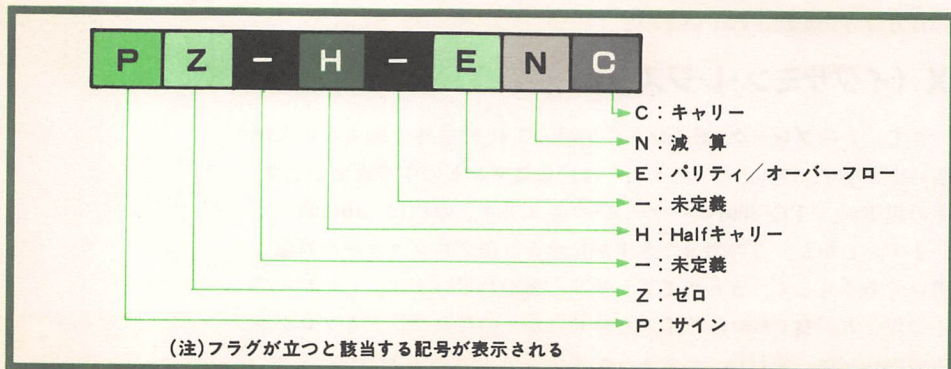
【図1·13】——Xコマンドの起動

とキーインしてください。図1・13のように表示されるでしょう。これは、Xコマンドを起動したものです。ブレーク・ポイントは、このXコマンドと併用すると効果的です。

X ⏎

とキーインすることで、全レジスタの値を知ることができます。記号の見方は、次のとおりです。

【図1・14】——Xコマンドによる表示

また、フラグに用いられている記号は、次のとおりです。

【図1・15】——フラグの記号

ちなみにXコマンドは、

X（イグザミン・レジスタ：examine register)

の意味です。

Xの用法——その2

Xコマンドは、レジスタの値を表示するだけでなく、

レジスタの値を変更する

のにも使います。ブレーク・ポイントの実験をするため、ここでXコマンドを用いて、Aレジスタの値を変更しておきます。

```
XA ⏎
```
Aレジスタの変更を意味します

とキーインしてください。次のようにAレジスタの現在の値が表示され、入力待ちとなります。

```
h]XA
A :00=■
```
入力待ち

【図1・15】——Aレジスタを指定

ここでは、

Aレジスタ＝９９H

に変更しますので

```
99 ⏎
```

とキーインします。

```
h]XA
A :00=99
h]■
```
コマンド・レベルに戻る

【図116】——Aレジスタ＝９９Hに変更

そして、再び全レジスタを表示します。

```
X ⏎
```

とキーインしてください。

Aレジスタ＝９９H

に変更されたのが確認できます。

```
h]X
A :99 F :PZ---E-- B :0000 D :EDCC H :0001 A':00 F':PZ---E-- B':0000 D':EDCC
H':0001 IX:1340 IY:0000 I :F3 PC:0000 SP:E5F5
h]■
```
Aレジスタ＝９９Hに変更されている

【図1・17】——確認

実験用プログラム

ここからブレーク・ポイントの実験に入ります。

まず、図1・18のプログラムをキーインしてください。このプログラムを逆アセンブルしたのが、図1・19です。プログラムの意味を図の番号順に見ますと、

```
h]DC000,C004
C000 3E 01 3D 3D 76
h]■
```

【図1・18】――実験用プログラム

```
h]LC000,C004                                 ザイログ表記
①C000   3E 01        MVI  A,01      [LD   A,01]
②C002   3D           DCR  A         [DEC  A   ]
③C003   3D           DCR  A         [DEC  A   ]
④C004   76           HLT            [HALT     ]
h]■
```

【図1・19】――逆アセンブル・リスト

❶　Aレジスタに01Hを代入

図1・16のように現在、Aレジスタの値は99Hになっています。しかし、❶により

Aレジスタ＝01H

に変化するはずです。

❷　Aレジスタから1を減算

Aレジスタ＝01H－01H＝00H

になるはずです。

❸　Aレジスタから1を減算

Aレジスタ＝00H－01H＝FFH

になるはずです。

❹　プログラムの実行停止

ブレーク・ポイントを設定

さて、以上の各ステップで、プログラムの途中経過を調べたいとしたら、どうしたら良いでしょう？

そこで登場するのが、

ブレーク・ポイント（break point）

です。まず、❶で

Aレジスタ：99H――→01H 〔実行前〕　〔実行後〕

と変化する様子を確認してみましょう。それには、プログラムの実行開始後、

❷の直前＝C002H

でプログラムの実行をSTOP（ブレークをかける）をしてやれば良いのです。

```
h]GC000,C002
```

(C000: 実行開始アドレス、C002: ブレーク・ポイント・アドレス)

とキーインしてください。次のように、マシン語モニタのコマンド待ちとなります。

```
h]GC000,C002
h]■
```

【図1・20】――ブレーク・ポイントを設定

そして、Xコマンドを実行します。

X ⏎

とキーインしてください。

```
h]X
A :01 F :M----ON- B :0000 D :EDCC H :0001 A´:FF F´:PZ---E-- B´:0000 D´:EDCC
H´:0001 IX:1340 IY:0000 I :F3 PC:C002 SP:E5F1
h]■
```

(Aレジスタ＝０１　に変化／Ｃ００２Ｈでブレークがかかった)

【図1・21】――Xコマンドで確認

ご覧のように

PC（プログラム・カンター）＝Ｃ００２Ｈ

でブレークがかかったことがわかります。そして、

Aレジスタ＝０１Ｈ

にセットされたこともわかります。

続いて、❷の部分を調べましょう。それには、

❸＝Ｃ００３Ｈ

実行直前にブレークをかければ良いのです。

```
h]GC000,C003
h]X
A :00 F :M----ON- B :0000 D :EDCC H :0001 A´:01 F´:M----ON- B´:0000 D´:EDCC
H´:0001 IX:1340 IY:0000 I :F3 PC:C003 SP:E5F1
h]■
```

(Ｃ００３Ｈにブレーク・ポイントを設定／Aレジスタ＝００Ｈに変化／ブレーク・ポイント・アドレス)

【図1・22】――Ｃ００３Ｈでブレークをかける

同様にして、❸の実行結果を調べるには、

❹＝Ｃ００４Ｈ

でブレークをかけます。次の図で確認してください。

```
h]GC000,C004 ◀── C004Hにブレーク・ポイントを設定
h]X          ──── Aレジスタ＝FFHに変化
A :FF F :M--H-ON- B :0000 D :EDCC H :0001 A':00 F':PZ---ON- B':0000 D':EDCC
H':0001 IX:1340 IY:0000 I :F3 PC:C004 SP:E5F1
h]■
                               └── ブレーク・ポイント・アドレス
```

【図1・23】――Ｃ００４Ｈでブレークをかける

2か所のブレーク・ポイント・アドレス

以上が、ブレーク・ポイントの基本的な使い方です。まとめますと、

> ［ブレーク・ポイントの使い方］
>
> ●プログラムの途中、実行を停止したい位置にブレーク・ポイント・アドレスを設定する（Ｇコマンド）。
>
> ●プログラムが停止したら、各レジスタの値を調べる（Ｘコマンド）

ブレーク・ポイントを有効に使うことで、デバッグの強力な武器となります。

さて、PC-8801のマシン語モニタでは、

ブレーク・ポイントを２か所設定可能

となっています。その使い方を説明しておきましょう。次のプログラムで実験してみます。

```
h]DC000,C007
C000 3E 10 FE 11 38 01 76 76
h]■
```

【図1・24】――第２ブレーク・ポイント実験用プログラム

```
  h]LC000,C007                              ザイログ表記
❶ C000   3E 10         MVI  A,10         [LD    A,10 ]
❷ C002   FE 11         CPI  11           [CP    11   ]
❸ C004   38 01         JRC  C007         [JR    C007 ]
❹ C006   76            HLT               [HALT       ]
❺ C007   76            HLT               [HALT       ]
❻ h]■
```

【図1・25】――逆アセンブル・リスト

図1・25が、その逆アセンブル・リストです。図の番号でプログラムの流れを考えますと、

❶　Aレジスタに10Hをセット。

❷　Aレジスタと11Hを比較。

❸　比較の結果、

CY（キァリー・フラグ）＝1

ならば、❺へジャンプ。

❹　比較の結果、CY＝0なら、プログラムはここでSTOPする。

❺　比較の結果、CY＝1なら、プログラムはここでSTOPする。

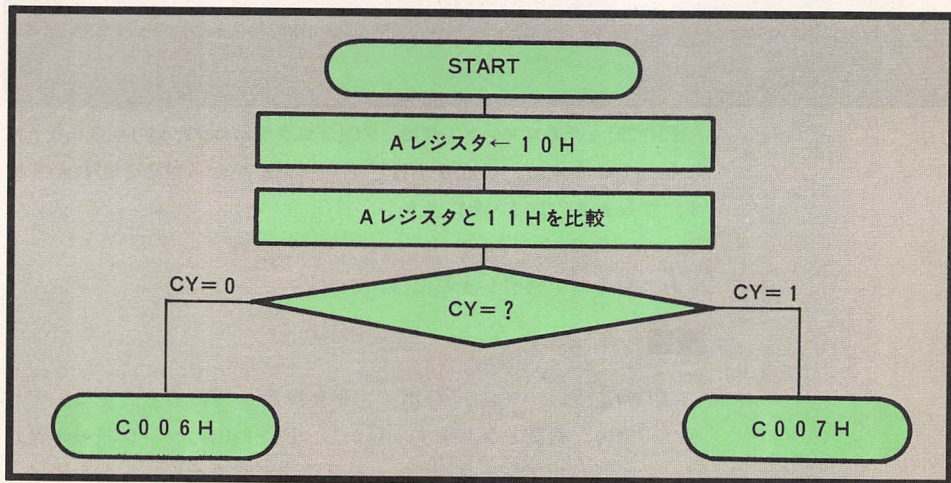
さて、基本的な問題ですが、このプログラムで❷が実行された後、CYの値は0と1のどちらになるでしょう？　それによって❸以下の分岐先が変化します。

第2ブレーク・ポイント・アドレス

この例は、比較的単純なケースですが、複雑なプログラムになりますと、分岐の仕方が難しくなります。そしてデバッグの際、どちらに分岐するか調べてみたいことが起こります。そんな時、

ブレーク・ポイントを複数個設定する

方法が用いられます。

上の例では、次のように行います。次のフロー・チャートをご覧ください。

【図1・26】──フロー・チャート

このフロー・チャートによれば、CYの値により、

CY＝0　→C006H

CY＝1　→C007H

のいずれかでプログラムがSTOPすることがわかります。そこで、

C006H――第1ブレーク・ポイント・アドレス C007H――第2ブレーク・ポイント・アドレス

の2か所にブレーク・ポイントを設定してやります。次のようにキーインしてください。

```
GC000,C006,C007
```

GC000 ―― 実行開始アドレス
C006 ―― 第1ブレーク・ポイント・アドレス
C007 ―― 第2ブレーク・ポイント・アドレス

次のようにブレークがかかり、プログラムの実行が停止します。

```
h]GC000,C006,C007
h]■
```

【図1・27】――2か所のブレーク・ポイントを設定

そうしましたら、

X ⏎

でレジスタ類を表示させます。

```
h]X
A :10 F :M--H-ONC B :0000 D :EDCC H :0001 A´:00 F´:PZ---E-- B´:0000 D´:EDCC
H´:0001 IX:1340 IY:0000 I :F3 PC:C007 SP:E5F1
h]■
```

PC:C007 ―― C007Hでブレーク

【図1・28】――Xコマンドで確認。PC(プログラム・カウンタ)＝C007Hになっていますね。C007Hでブレークがかかったことがわかります。すなわち、図1・26により

CY＝1

であったことが判明しました。

課題

以上のように、ブレーク・ポイントを複数か所設けることは、デバッグの強力な武器となります。しかし、PC-8801のマシン語モニタでは、2か所しかブレーク・ポイントを設定できません。そんな時は、どうしたら良いのでしょう？　何か良い方法はないのでしょうか？　マア、次の章へお進みください。

第2章

ブレーク・ポイント処理とマシン語モニタのホット・スタート

2

本章では前章に引き続き、マシン語モニタの解析をもとに、

マシン語プログラマーのための

技術情報

を探って行きます。

最初は、マシン語プログラマ必須の

モニタ・ホット・スタートへの道

を探ります。これがなかなかの曲者で、N-BASICマシン語モニタのように

JP 〈ホット・スタート〉

では片付きません。

ダンプ・レジスタ

ROMのバンク切り換え

の付随機能のためです。

次いで前章でマスターしたブレーク・ポイントの活用法をもとに

ブレーク・ポイント設定のしくみ

を探って行きます。そして、その結果として

●マシン語プログラムの合理的な止め方

●複数か所のブレーク・ポイント設定法

が、そのしくみと合わせてわかります。マシン語プログラマー必須の技術情報――それは自分でマシンを解析して初めて得られるのです。

2・0　マシン語モニタ、ホット・スタートへの道

突然ですが、話は抜粋から始まります。

「(HALTを使うと) 確かにプログラムは実行を停止します。しかしその後は、どのキーを押しても (ストップ・キーでも)、ウンともスンとも言わなくなります。これを止めるにはリセット・キーしかありません。

プログラムを走らせる度にこんな手間をかけるのはめんどうです。

以上のわずらわしさを体験してもらったあと、マシン語のプログラムを止めるには、

HALT ――――――――×
JP　　5C66H――――○

の方法を示しました。つまりモニタのスタート・アドレスに飛ばせる方法です。これなら一発でプログラムを止めたあと、モニタ・コマンド・レベルに戻ります」(『PC-8001マシン語入門』電波新聞社、P.169)

マシン語モニタのホット・スタート

以上は、私が某マイコン・クラブで行なった「マシン語講座」をまとめた記事の一節です。普通、マシン語の入門書をひもときますと、

マシン語のプログラムを止める
→HALT命令を使う

と教えています。しかし、パソコン・ユーザーが実際にこの方法を用いますと、非常に不便な思いをします。そこで、私がPC-8001のユーザーを対象に行なったマシン語講座では、

JP　　5C66H――――Ⓐ

という方法をお勧めしました。プログラムの最後にこの命令を書いておきますと、自動的にマシン語モニタに戻りますから、非常に便利なのです。

Ⓐの方法は、非常に便利です。しかし、この方法はPC-8001のユーザーしか使えません。それなら、我々PC-8801のユーザーは、どうしたら良いのでしょう。

<PC-8001>
マシン語モニタのホット・スタート＝５Ｃ６６Ｈ

このことと、Ⓐを比べてみてください。マシン語のプログラムを止めるには、

JP 〈マシン語モニタのホット・スタート〉

とすれば良さそうだ、と想像がつきます。

JP 6047H?

それなら、

PC-8801のマシン語モニタの

ホット・スタートは何番地？

ということになります。

結論を申し上げましょう。

〈PC-8801〉

マシン語モニタのホット・スタート・アドレス＝6047H

です。

［疑問——その1］

PC-8801のホット・スタート＝6047H

は、どのようにして調べたのか？

マア、それはよしとしておいて、さっそくPC-8801でも、きれいにマシン語のプロダラムを止めてみましょう。

JP 6047H

とする？

残念でした。これだけでは、正解の半分です。100点満点の50点です。なぜでしょう？

メモリ・バンクの問題

我々が今まで問題にしてきた中で、最も大きな

PC-8001 ←→ PC-8801

の違いは？

——YES. PC-8801には、

メモリ・バンクの問題が存在する

ということでした。PC-8801のマシン語モニタは？

——YES. N-BASIC ROMの後方に収められていました。しからば、マシン語モニタをホット・スタートさせるには、単純に

JP 6047H

としたのではダメなことは、おわかりでしょう。

ROMの選択を正しく

PC-8801で、マシン語モニタをホット・スタートさせるには、次の

手順を踏めば良さそうだと予想されます。

●N-BASIC ROMに切り換える
●JP　6047H　を実行する

【図2・0】――ホット・スタートの手順

それでは、N-BASIC ROMに切り換えるにはどうしたら良いでしょうか？

最も簡単なのは、RAM上に置かれた次のシステム・サブルーチンをCALLすることです。

[N-BASIC ROMの選択]
アドレス　F16CH
機能
このサブルーチンをCALLすると、今までのモードに無関係にN-BASIC ROMのモードに切り換わる。

これを利用しますと、

マシン語のプログラムの実行を止める
＝マシン語モニタをホット・スタートさせる

には、次のようにすれば良さそうですね？

```
CALL  F16CH  ――N-BASIC ROMへ切り換え
JP    6047H  ――ホット・スタートへ
```

【図2・1】――ROMを切り換えてから

JP→CALL

実は、これでは正解と言えません。半分位は合っていますが――。

実験してみればすぐにわかりますが、確かに図2・1の方法でもマシン語のコマンド・レベルに戻ります。しかし、コントロールBを実行してもBASICに戻れなくなります。そして、(その時のHLレジスタの値にもよりますが) 大ていは暴走してしまいます。

なぜでしょう？

正しい方法は、次のとおりです。

```
CALL  F16CH
CALL  6047H
└──┘
  └――JPがCALLに置き換えられたことに注意！
```

【図2・2】――モニター・ホット・スタートの正解

それでは、実際に実験してみます。図2・2をアセンブル・リストで示したのが、次の図2・3です。そして、これにしたがって6バイト分のマシン語を入力し、Dコマンドで確認したのが図2・4です。

```
                       ;====================================
                       ;  MONITOR HOT START:1983.5.24
                       ;------------------------------------
                       ;
                                ORG   0BB00H
                       ;
6047                   MHOT:    EQU   6047H
F16C                   NROM:    EQU   0F16CH
                       ;
BB00 CD6CF1                     CALL  NROM                 ;SELECT N-BASIC ROM
BB03 CD4760                     CALL  MHOT                 ;JUMP TO MONITOR
                       ;
                                END
```

【図2・3】――アセンブル・リスト

```
h]DBB00,BB05
BB00 CD 6C F1 CD 47 60
h]■
```

【図2・4】――ダンプ・リスト

入力が済みましたら

GBB00 ⏎

でスタートさせます。

```
h]GBB00
h]■
```

【図2・5】――プログラム実行

ご覧のようにマシン語のコマンド・レベルに達しました。ためしに

CTRL + **B** （コントロールB）

を押してみます。

```
h]GBB00
h]^b
Ok
■
```

【図2・6】――コントロールB

うまく、N88-BASICに戻ることができました。

2·1 ブレーク・ポイント処理のしくみ

またまた我々は、新しい課題をかかえ込むことになりました。

●なぜ、PC-8801マシン語モニタの

ホット・スタート・アドレス＝６０４７Ｈ

なのか？

●なぜ、マシン語のプログラムをSTOPさせるのに

```
CALL F16CH
CALL 6047H
```

としなければならないのか？　他にもっと簡単な方法はないのか？

さらに言えば、前章末において、次のような課題も残っています。

●３つ以上のブレーク・ポイント・アドレスを設定することは不可能か？　もしそれが可能なら、どのように行うのか？

実は、これらの課題はマシン語モニタの中の

ブレーク・ポイント処理のしくみ

を解析することで解決してしまいます。ブレーク・ポイントの使い方は、すでに１・２節でやりました。あとは、ブレーク・ポイント処理のしくみを解析して行くのみです。

ブレーク・ポイント設定のしくみ

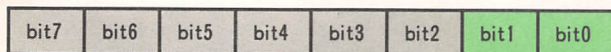
まず、ブレーク・ポイントが設定される時、モニタ内部でどんな処理がされているかを探ってみます。ブレーク・ポイントは、ユーザーがＧコマンドを実行した際に設定されます。ですからＧコマンドを解析した１・０節の部分を機械語レベルで調べていけば良いのです。

ブレーク・ポイントが設定されると、次の３つの処理が実行されます。

❶ フラグの設定（Ｆ１ＣＦＨ）

これについては、図１・６で説明いたしました。次の図２・７にまとめておきます。設定の順序は、次のとおりです。

６４１２Ｈ：フラグのクリア（００Ｈの書き込み）
６４３ＣＨ：第２ブレーク・ポイント設定のフラグを立てる
６４４ＥＨ：第１ブレーク・ポイント設定のフラグを立てる

この結果、図2・7のようにフラグがセットされ、ここを調べることで、ブレーク・ポイントの有り・無しがわかります。

【図2・7】――ブレーク・ポイント設定フラグ

❷ 情報のストア

この部分は、Gコマンド解析の時には省略した部分です。ここでいう情報とは、

ブレーク・ポイント・アドレス

ブレーク・ポイント・アドレスのマシン・コード

の2点です。これらをシステム・ワーク・エリアにストアします。なぜこれが必要かは、次の❸をご覧いただければおわかりになるでしょう。

	第1ブレーク・ポイント	第2ブレーク・ポイント
アドレス	F1D4H～F1D5H	F1D1H～F1D2H
マシン・コード	F1D3H	F1D0H

【図2・8】――情報のストア

概当するシステム・ワーク・エリアは、図2・8のとおりです。

❸ リスタート命令の書き込み

ブレークをかけるアドレスに、

RST 38H (マシン・コード＝FFH)

を書き込みます。たとえば図2・9のプログラムでC002Hにブレーク・ポイントを設定するなら、

〔アドレス〕	〔マシン・コード〕
C000H	01H
C001H	02H
C002H	03H
C003H	04H
C004H	05H
C005H	06H
⌠	

ブレーク・ポイント設定 → C002H＝FFH ← リスタート命令

【図2・9】――リスタート命令の書き込み

C002H＝FFH

に書き換えます。

まてよ、――。

そんなことをしたら、元のプログラムがわからなくなってしまうのでは？　だいじょうぶです。そのために❷において、システム・ワークエリアに

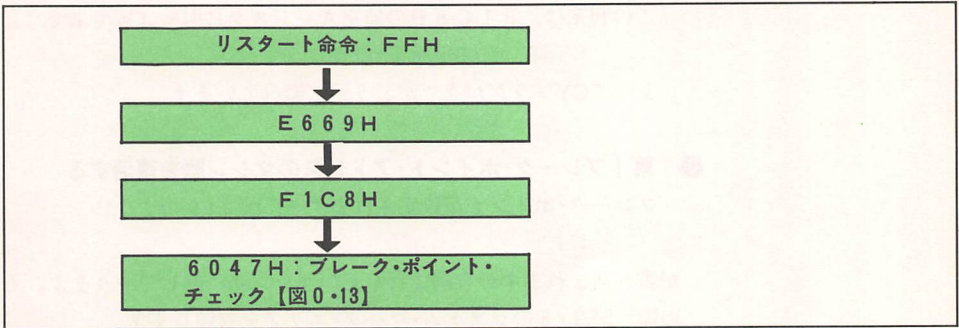
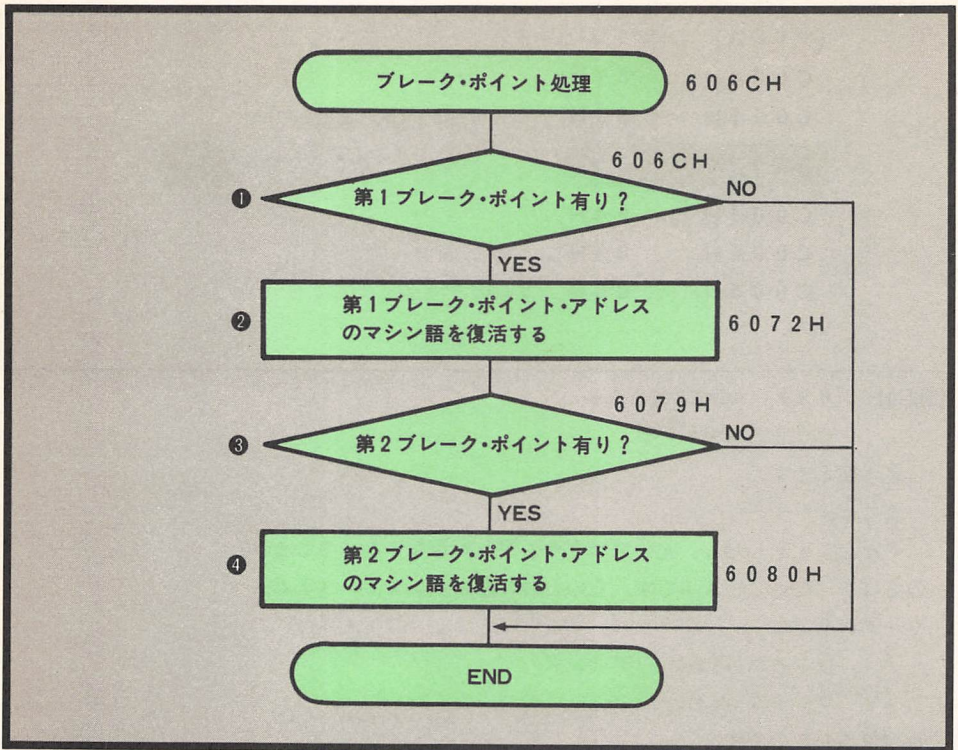
ブレーク・ポイント・アドレス

ブレーク・ポイント・アドレスのマシン・コード

を記憶させたのです。

ブレーク・ポイント実行後の処理

ブレーク・ポイントが設定されると、以上の三つの処理が実行されます。そして、Gコマンドの残りの処理により、ユーザー・プログラムが走り出します。

さて、ユーザー・プログラム実行中に**ブレーク・ポイント**に到達しますと、

の過程を経て、6047Hにジャンプしてきます。そして、図2・7のように、

【図2・10】――ブレーク・ポイント処理

ブレーク・ポイント設定フラグ＝F1CFH

の値を見て、ブレーク・ポイントが設定されていると、次のブレーク・ポイント処理を行ないます。

上のフローチャートの番号にしたがって、説明していきます。

❶ 第1ブレーク・ポイントのチェック

この判定は、F1C8Hの値をAレジスタに引っぱって来て

RRCA：bit 0――►CYフラグ

によってCYフラグが立つかどうかで調べています。

❷ 第1ブレーク・ポイント・アドレスのマシン語を復活する

ブレーク・ポイントが設定されますと、図2・9のように

FFH

が書き込まれますから、これを元のマシン語に直してやります。その際、図2・8のシステム・ワーク・エリアを参照します。

❸ 第2ブレーク・ポイントのチェック

❶とは異なり、AND命令によるZフラグの変化で調べています。

❹ 第2ブレーク・ポイント・アドレスのマシン語を復活する

❷と同様の処理を行います。

ブレーク・ポイント処理の要

いかがですか？　以上が、ブレーク・ポイント処理のすべてです。これを図式化すると、次の図2・11のようになります。これを見ますと、どうやらブレーク・ポイント処理の要は、

リスタート命令

RST　38H（マシン・コード＝FFH）

にあるのが、おわかりでしょう。

●マシン語プログラムの実行を、合理的にSTOPする。
●3つ以上のブレーク・ポイントを設定する。

これら各々の方法は、

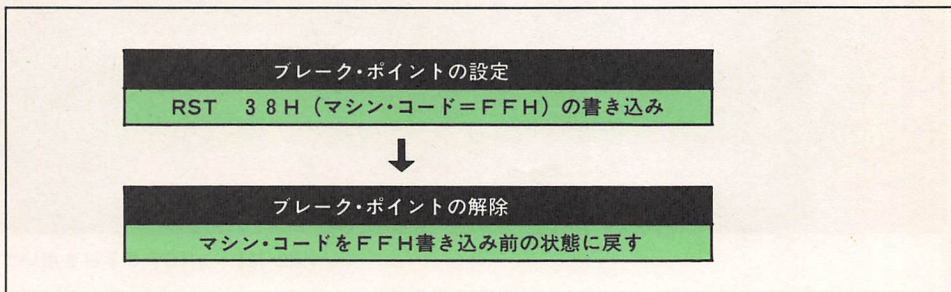
リスタート命令――RST　38H

にあるのは、もう見え見えです。そこら辺を、次節において解明してまいりましょう。

ブレーク・ポイントの設定

RST　38H（マシン・コード＝FFH）の書き込み

ブレーク・ポイントの解除

マシン・コードをFFH書き込み前の状態に戻す

【図2・11】――ブレーク・ポイント処理のしくみ

2·2 RST 38Hの威力

RST ３８H（マシン・コード＝FFH）

この命令が、マシン語の実行中、

プログラムを停止させる

ブレーク・ポイントをかける

上で重要な役割を果たしそうなことがわかってきました。それではその具体的な使い方は？　次をご覧ください。

リスタート命令を使って

まず、マシン語プログラムの止め方です。

［秘　伝］
マシン語のプログラムを停止するには
RST ３８H（マシン・コード＝FFH）
を用いれば良い。

実験してみます。

まず、次のプログラムを入力してください。

【図2・12】――RST ３８Hを用いて

マシン語は、C０００Hから、

３EH　８８H　FFH

と３バイト入力していただければ結構です。プログラムの意味は、次のとおりです。

❶ Aレジスタに８８Hをセットする。

❷ プログラムをSTOPするため、リスタート命令を書き込む。

ここで注意していただきたいのは、インテル形式とザイログ形式で、リスタート命令のオペランドが異なるということです。同じマシン・コードFFHを書き込むにしても、

```
インテル形式：RST　7
ザイログ形式：RST　38H
```
マシン・コードは同じFFH

のように異なります。次にリスタート命令の一覧を掲げておきます。

インテル形式	ザイログ形式	マシン・コード
RST　0	RST　0H	C7H
RST　1	RST　8H	CFH
RST　2	RST　10H	D7H
RST　3	RST　18H	DFH
RST　4	RST　20H	E7H
RST　5	RST　28H	EFH
RST　6	RST　30H	F7H
RST　7	RST　38H	FFH

【図2・13】――リスタート命令一覧表

プログラムの停止に成功

プログラムの入力が終りましたら、

GC000 ⏎

でプログラムを実行します。ご覧のように、ただちに実行が終了し、コマンド待ちとなります。

```
h]GC000
h]■
```

【図2・14】――プログラムの実行が停止する

そして、

X ⏎

を実行します。

```
h]X
A :88 F :PZ---E-- B :0000 D :EDCC H :0001 A':88 F':PZ---E-- B':0000 D':EDCC
H':0001 IX:E400 IY:06F3 I :F3 PC:C002 SP:C4E9
h]■
```
Aレジスタ＝88Hになっている
C002HでSTOP

【図2・15】――Xコマンドで調べる

ご覧のように

Aレジスタ＝88H

PC＝C002H

になっているのが確認できます。

2・3 技術情報発見のルーツ

前節においては、マシン語のプログラムを停止させるには、

RST　３８H（マシン・コード＝ＦＦH）

を用いれば良いことを、実験でたしかめました。本節では、さらにつっ込んだ考察を試みます。そして、なぜＦＦＨを書き込めば良いのかを、マシン語モニタのしくみとからめて探って行きます。

RST　38H と CALL　38H

ところで、マシン語レベルで見ると、

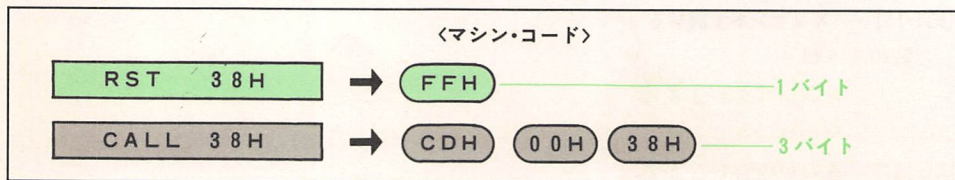
RST　38H

とは、どんな命令でしょうか？

【図2・16】――RST　３８Hの働き

RST　３８Hの働きは、上の図のようになっています。すなわち、現在のプログラム・カウンターの値をスタック領域にストアし、プログラム・カウンターの値を００３８Hにします。これは、とにもかくにも

CALL　38H

とまったく同じ、ということになります。ただし、マシン・コード・レベルで見ますと、次のように〈RST　３８H〉の方が、省エネであるといえます。

JP 0000Hの怪

以上のように、<RST 38H> を用いますと、

CALL 38H

が実行されますから、プログラムの制御は、

38Hへジャンプ

することになります。

それでは、38Hにはどんなプログラムが書かれているでしょう？

L38,3A ⏎

とキーインして、逆アセンブルしてみてください。

```
hJL38,3A                                    ザイログ表記
0038    C3 E669        JMP   E669         [JP     E669]
hJ■
```

【図2・17】――L38, 3 Aの実行

ご覧のようにE669Hにジャンプしていることがわかります。そこで、

LE669,E66B ⏎

とキーインしてみてください。

```
hJLE669,E66B                                ザイログ表記
E669    C3 0000        JMP   0000         [JP     0000]
hJ■
```

【図2・18】――LE669, E66Bの実行

この出力画面を良くご覧ください。何と

0000Hにジャンプ

しているのがわかります。これはおかしいですね？

(注)もし出力画面がこのようにならない時は、もう一度電源を入れなおし、メモリをクリアしてから実験してみてください。

E669H〜E66BHの変化

この疑問は、次の実験を試みることで解決します。

まず、次のようにC000HにFFHを書き込んでください。

```
hJSC000
C000 C3-FF
hJ■
```

【図2・19】――C000HにFFHをセット

そして、

GC000H ⏎

でプログラムを走らせます。もちろん、FFHが書き込まれていますから、すぐにプログラムは止まります。

```
h]GC000
h]■
```

ブレークがかかり、コマンド待ちに

【図2・20】——プログラム終了

ここまでは、どうということはありません。さて、ここで図2・18でやりました

LE669,E66B ⏎

をもう一度実行します。そして、実行結果を比べてみてください。

```
h]LE669,E66B
E669    C3 F1C8       JMP   F1C8        [JP    F1C8]
```

ザイログ表記

【図2・21】——0000HがF1C8Hに変化

何と、ご覧のように変化しています。先ほどは、

JP 0000H

だったものが、今度は、

JP F1C8H

になっています。

ジャンプ・テーブル書き換えのルーツ

これは、一体いつ変化したのでしょう？　図2・18と図2・21の間には、Gコマンドを実行しています。とすると、

犯人はGコマンド？

ということになります。

もちろん、その通りです。Gコマンド処理ルーチンの中で

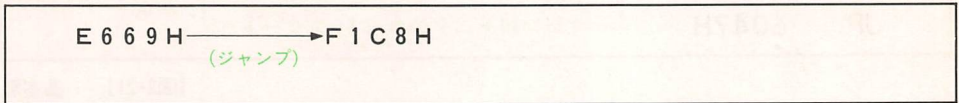
E66AH～E66BH

が変化しています。それが、第1章では説明を省略しておいた図1・2における❷—(1)の部分です。もう一度、図1・1を良くご覧になってください。今やこの意味がよくおわかりになったことと思います。

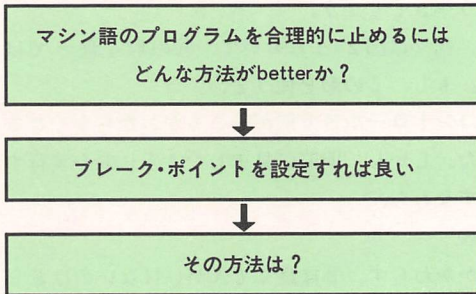
技術情報発見のルーツ

Gコマンド処理ルーチンによるジャンプ・テーブルの書き換えがわかったところで先に進みます。

図2・21により、

E669H ──(ジャンプ)──▶ F1C8H

することがわかりましたので、今度はF1C8Hを調べてみます。

LF1C8,F1CD ⏎

とキーインしてください。図2・20のように出力されるでしょう。何と、図2・1と同じ画面が得られました。

我々が、今まで実験してきたこれらの方法は、RST 38H、すなわち、マシン語モニタ（とりわけGコマンド処理）のしくみを解析することで発見できたのです。

【図2・20】――プログラムの実行停止法の出現

2・4 ユーザー・スタック・エリアを探る

ついに我々は、マシン語プログラムの停止方法である

CALL 61FCH	N-BASIC ROMへの切り換え	Ⓐ
JP 6047H	マシン語モニタのホット・スタートへ	

【図2・21】——基本形

のルーツを発見しました。これは、**6バイトのマシン語**です。これと同じことを、リスタート命令を使えば、

RST 38H ——Ⓑ

と、たった1バイトで実現することができます。ですから、実用上はⒷの形を使うべきでしょう。

ところで、我々は図2・1において、Ⓐの形を使ってはいけないとされました。もし、この形を使うと、

コントロールBで何が起こるかわからなくなる!

とされました。しかし、前節で見ましたようにⒷを実行するとⒶが起こります。すなわち、

Ⓐ ＝ Ⓑ

です。にもかかわらず、Ⓑは良くてⒶがいけないのはどうしてでしょうか?

ユーザー・スタック・エリアのしくみ

ⒶもⒷも見かけ上は、まったく同じことを実行します。しかし、たった1か所、異なる点があります。それは、

スタック・ポインタの位置

です。次ページの図をご覧ください。

MON ⏎

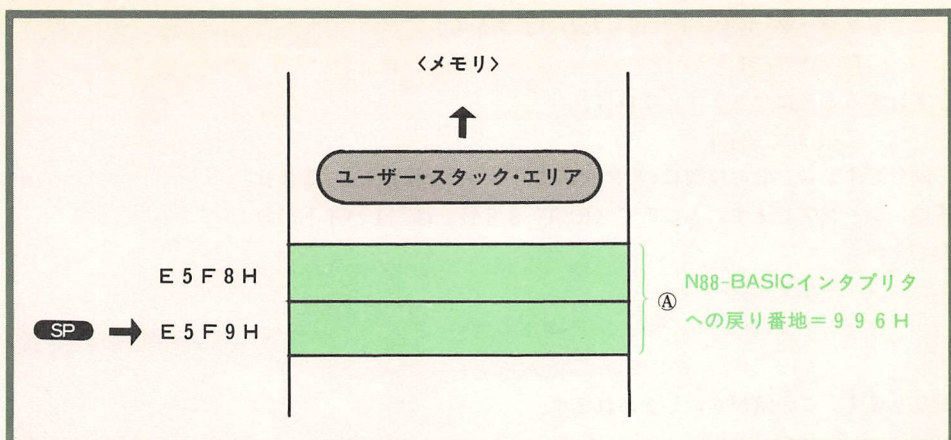
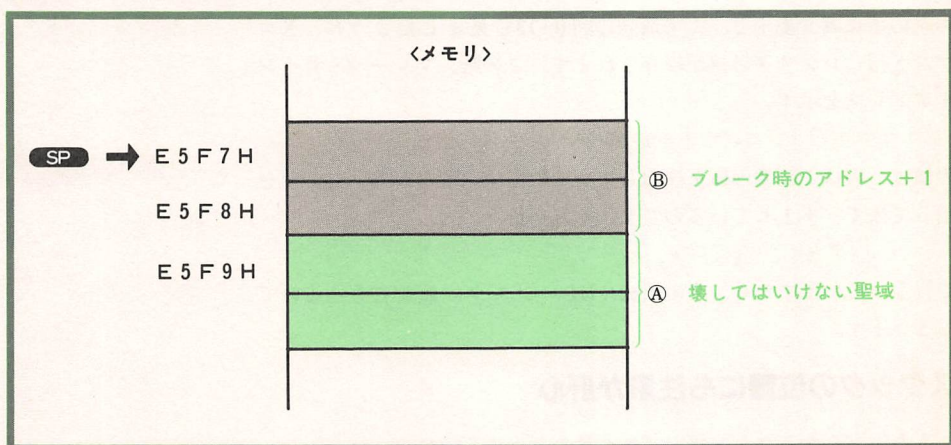
でマシン語モニタに飛び込んだ時の、SP(スタック・ポインタ)の状態を表わしたのが、図2・21です。Ⓐの領域に

BASICへの戻り番地＝996H

が保存されています。そして、SPは図の位置にあります。

(注) この図から、N88-BASICインタプリタへ戻るには、単純に

RET(マシン・コード＝C9H)

を実行するだけで良いことが、おわかりでしょう。

【図2・21】――マシン語モニタ、メモリ・マップ

【図2・22】――ブレーク時のメモリ・マップ

さて、マシン語モニタでは、

Ⓐより手前の領域＝E5F8H以前の領域

が、**ユーザー・スタック・エリア**となります。ユーザー・プログラムの実行開始とともに、SPの値が変化して行きます。そして、スタックが深くなるにつれて、SPの値も若くなって行きます。プログラムの進行とともに、SPの値は上下します。やがて

ブレーク・ポイント

に到達した時、SPが元の位置に戻っていたとします。すなわち

SP＝E5F9H

に回復していたとします。

ここから、説明は次の図2・22に移ります。

ここでブレーク・ポイントに達したわけですから、

RST 38H

を実行することになります。これは、

CALL 38H

と同じですから、Ⓑの位置に〈リターン・アドレス〉を保存して、38H番地にジャンプします。ところで〈RST　38H〉は、1バイトの命令ですから、

〈リターン・アドレス〉
=〈ブレーク・ポイント・アドレス〉－1

となります。この値がⒷにしまわれます。

さて、ブレークがかかり、プログラムの制御がふたたびマシン語モニタの手に渡りますと、まず最初に図0・13で見ましたように、Xコマンド用にレジスタの値が保存されます。この時、ブレーク・ポイントアドレスを示す。

PC（プログラム・カウンタ）

の値は、実はこのⒷの値を採用しているのです。すなわち、Ⓑの値を拾って来て、＋1しているのです。それを拾うのに、

EX　(SP),HL

で行なっていますから、Ⓑの領域にHLレジスタの値が書き込まれてしまいます。

スタックの位置にも注意が肝心

さあ、そろそろマシン語のプログラムを止めるのに、

```
CALL F16CH
JP   6047H    ———❶
```

はダメで、

RST　38H

ならOKである理由がわかってきたのではないでしょうか？

もし、このようにすると、

マシン語モニタのホット・スタート＝6047H

に飛んだ時に、

SP＝E5F9H（図2・21の位置）

を指すことになります。するとアホなモニタはⒶの部分がブレーク・ポイント・アドレスと思い、そこにHLレジスタの値を書き込んでしま

います。すなわち、

BASICへの戻り番地が破壊されてしまう!

ことになります。

以上が、❶を行ってはいけない理由です。もちろん、図2・22のように、

```
CALL F16CH
CALL 6047H
```

とするのはかまいません。

注意1

マシン語モニタの手にプログラムの制御が移った後、Ⓑの領域には、

エラー処理ルーチンの入口=60E0H

が書き込まれます(図0・14の❽)。

注意2

ユーザー・プログラムの中でPOP命令等でスタックを浅くしたところでブレークをかけると、当然Ⓐの領域はこわされてしまいます。

注意3

ブレーク後は、その時のSPの値が保存されています。したがってその後、Gコマンドを実行しますとSPはその位置から再開します。すなわち、SPの値は浮動的です。ここがN88-BASICのマシン語モニタと、N-BASICマシン語モニタの大きな違いです。

注意4

Ⓐの部分にHLレジスタの値が書き込まれますと、コントロールBでそのアドレスにジャンプしてしまいます。

2·5 手動ブレーク・ポイントの設定

前節末の注意、とりわけ 注意 2 注意 3 などはマシン語を扱う上で注意する必要があります。こうした事項は、マニュアルに記載されていても良さそうな感じがしますが、実際は説明されていません。現状では、やはりユーザーが自分の手でROMを解析して行くしか手がないようです。

さて、本章の最後に、

3か所以上のブレーク・ポイントを設定

する方法を検討してみましょう。このことは、すでに〈ブレーク・ポイント設定のしくみ〉を知った我々にとっては、明らかなことかもしれません。

キャラクタ・コードを判定する

まず、この実験に用いるプログラムです。次がそれで、入力キャラクタにより、3か所に分岐するように作ってあります。最初に、簡単にこのプログラムを説明しておきます。図の番号を参照しながら、どうぞ。

```
                    ;=====================
                    ;  MULTI BREAK POINT
                    ;           1983.6.5
                    ;=====================
                    ;
  0018              PR:     EQU  18H
  3583              INPUT:  EQU  3583H
                    ;
                            ORG  0BB00H
                    ;
① BB00 3E0A         EX:     LD   A,0AH
  BB02 CD1800               CALL PR              ;LINE FEED
② BB05 CD8355               CALL INPUT           ;INPUT A
③ BB08 CD1800               CALL PR              ;PRINT A
④ BB0B FE30                 CP   '0'
  BB0D 3805                 JR   C,E1            ;IF A<"0" THEN E1
⑤ BB0F FE3A                 CP   '9'+1
  BB11 3002                 JR   NC,E2           ;IF A>"9" THEN E2
⑥ BB13 76                   HALT                 ;"0"<=A<="9"
⑦ BB14 76            E1     HALT
⑧ BB15 76            E2:    HALT
                     ;
                            END
```

【図2・23】――3分岐を設定して

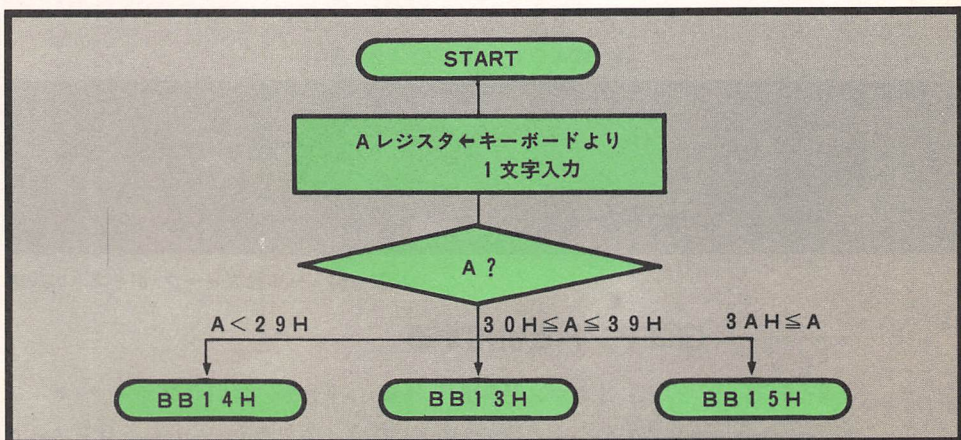
❶　１行改行します。
❷　キーボードから、１文字入力します。入力したキャラクタ・コードがＡレジスタにセットされます。
❸　入力したキャラクタ・コードをＴＶ画面に表示します。
❹　キャラクタ・コード〝0〞（３０Ｈ）と比較し、入力したキャラクタ・コードがこれより小の時は、ＢＢ１４Ｈにジャンプさせます。
❺　キャラクタ・コード〝9〞＋１（３ＡＨ）と比較し、それより大きい時は、ＢＢ１５Ｈにジャンプさせます。
❻　入力したキャラクタ・コードが、

３０Ｈ～３９Ｈ

の範囲にある時は、ここに飛び込みます。
❼　入力したキャラクタ・コードが、２９Ｈ以下の時は、ここに飛び込みます。
❽　入力したキャラクタ・コードが、３ＡＨ以上の時は、ここに飛び込みます。

以上をまとめますと、キーボードからＡレジスタに１文字入力し、

Ａ＜２９Ｈ　　→ＢＢ１４Ｈ
３０Ｈ≦Ａ３９Ｈ→ＢＢ１３Ｈ
３ＡＨ＜Ａ　　→ＢＢ１５Ｈ

ということになります。

【図2・24】──キャラクタ・コードで分岐

手動ブレーク・ポイントの設定

以上のように３か所に分岐するプログラムが、正しく動いているかをテストするためには、

```
BB13H ┐
BB14H ├ 3か所にブレーク・ポイントを設定
BB15H ┘
```

する必要があります。2か所までなら、マシン語モニタのGコマンドで設定できます。

それなら、残り1か所は——？

もちろん、そのアドレスに

```
RST   38H     (マシン・コード＝FFH)
```

を書き込めば良いのです。

やってみましょう。次が、図2・24を入力したところです。Dコマンドで確認しています。

```
h]DBB00,BB15
BB00 3E 0A CD 18 00 CD 83 35 CD 18 00 FE 30 38 05 FE
BB10 3A 30 02 76 76 76
h]■
```

【図2・25】——プログラムの入力確認

次に、3か所のうちの1か所、たとえば、

BB13H＝FFH

のように、FFHを書き込みます。これで、BB13Hにブレーク・ポイントが設定されたことになります。

```
h]SBB13
BB13 76-FF ◀——————— SコマンドでFFHを書き込む
h]■
```

【図2・26】——手動ブレーク・ポイントの設定

Gコマンドと併用する

以上、Sコマンドを用い、いわば手動で、強引にブレーク・ポイントを設定したことになります。残り、2か所？　これは、Gコマンドで設定できますね。走らせてみましょう。

```
GBB00,BB14,BB15
      └──┘ └──┘
       │     └──────────── 第3ブレーク・ポイント
       └────────────────── 第2ブレーク・ポイント
```

```
h]GBB00,BB14,BB15
■
```

【図2・27】――残り2か所の設定

ご覧のように、入力待ちとなります。まず、

＊（キャラクタ・コード＝2AH）

を入力してみましょう。

2AH＜30H

ですから、予想ではBB14Hに飛び込むはずです。

【図2・28】――＊の入力

ご覧のように、＊を入力した後、Xコマンドで各レジスタ類を表示させます。

A＝2AH

PC＝BB14H

に注目してください。プログラムは、第2ブレーク・ポイントで止まったことになります。

次に、同様にプログラムを走らせ、

5（キャラクタ・コード＝35H）

を入力してみます。

30H≦35H≦39H

```
h]GBB00,BB14,BB15
5
h]X
A :35 F :M--H-ONC B :0000 D :EDCC H :0001 A´:61 F´:P--H-ON- B´:0000 D´:BB16
H´:BE1B IX:CEF1 IY:D026 I :F3 PC:BB13 SP:B8FB
h]■
```

【図2・29】――5の入力

ですから、ＢＢ１３Ｈに飛び込むはずです。

図２・29で、実行結果を確認してください。第１ブレーク・ポイントでプログラムが停止したのがわかります。

最後に、もう一度プログラムを走らせます。そして、

Ａ（キャラクタ・コード＝４１Ｈ）

を入力してみます。

３ＡＨ≦４１Ｈ

ですから、下のように第３ブレーク・ポイントでプログラムが停止します。

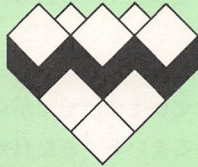
```
h]GBB00,BB14,BB15
A
h]X
A :41 F :P--H-ON- B :0000 D :EDCC H :0001 A':35 F':M--H-ONC B':0000 D':EDCC
H':0001 IX:CEF1 IY:D026 I :F3 PC:BB15 SP:B8FB
h]■
```

【図2・30】――Ａの入力

以上は、ブレーク・ポイントを３か所設定した例です。同様にして、

４か所、５か所、……

といくつでも設定することができます。せっかく見つけ出した手法です。おおいに活用し、マシン語プログラムのテストに活かしてください。

フック・アドレスと ROM別ダンプ

我々はPC-8801を扱うとき、すなわちプログラムを組むとき、

BASICインタプリタ

マシン語モニタ

のお世話になります。これらのシステム・プログラムは、

ROM

に格納されています。ですから、それが気に入ろうといるまいと、また自分用に改良を加えたくても、じっと我慢の子でいるしかありません。

本当でしょうか。

本章では、初めにこの問題にメスを入れます。そして、

フック・アドレスの機能

を明らかにします。

本章のもう1つの課題は、

モニタ・コマンド

です。何だ、今さらと思うかもしれません。我々は、普段何気なく、

Dコマンド

Lコマンド

などを単純に使っています。しかし、その中身を解析してみますと、いろいろなことがわかってきます。そして、自分でちょっと手を加えることで、

モニタ・コマンドの機能強化

をおこなうことができます。従来手の出なかったROMにもアクセスできるようになります。

いかにして、……。

まずは、次からお読みください。

3・0 フック・アドレスの活用

本書を手にされている方は、当然マシン語のプログラマーですから、かなりのレベルの方と予想されます。したがって、

〈ROM〉と〈RAM〉の違い

くらいは、よくご存知のことでしょう。決定的な違いは、

ROM——書き換え不能

RAM——書き換え可能

です。P C-8801のシステム部、すなわち、

N88-BASICインタプリタ

N88-BASICマシン語モニタ

は、ROMに収められています。したがって、その部分は（気に入らないところがあっても）書き換えられません。

ところが、……。

条件付きですが、ROM内に収められたシステム部でも、書き換えることが可能な場合があります。それには第0章、図0・11❸で説明を保留しておいた

フック・アドレス

を活用します。その使い方をこれから調べて行くことにいたします。

システムの書き換えに挑戦

何はともあれ、まずは実験をお目にかけましょう。そして、システムの書き換えを体験していただいた後、種明かしをいたします。

```
h]DBB00,BB49
BB00 21 07 BB 22 A0 ED FF 21 10 BB CD 50 55 C3 C1 4D
BB10 0A 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
BB20 2D 2D 0D 0A 20 20 B1 DD C0 20 C5 C6 20 D4 C2 C3
BB30 DD C9 3F 0D 0A 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
BB40 2D 2D 2D 2D 2D 2D 0D 0A 0A 00
h]■
```

【図3・0】——実験用プログラム

まず、このプログラムを入力してください。入力しましたら、

GBB00 ⏎

で走らせます。走らせても次の図のように何も起こりません。カーソルを表示してコマンド待ちになるだけです。

```
h]GBB00
h]■
```

何も起こらない

【図3・1】——プログラムの実行

続いて、コントロールBでBASICのコマンド・レベルに戻ります。これも、特に問題はありません。

```
h]GBB00
h]^b
Ok
■
```

コントロールB：〝さらばマシン語モニタよ〟

【図3・2】——コントロールBで

MONが書き換えられる

以上が準備です。

実は、図3・2が我々にとってマシン語モニタの見収めです。もう、2度とマシン語モニタには戻れません。

ウソか本当か、ためしてみればわかります。あなたは、マシン語モニタに入るのにどうしていますか？

MON ⏎

とキーインする？ OK!

MON ⏎

とキーインしてみてください。

【図3・3】——MONが使えなくなる

ご覧のように、ふざけたメッセージとともにベルが鳴り、

Feature not available エラー

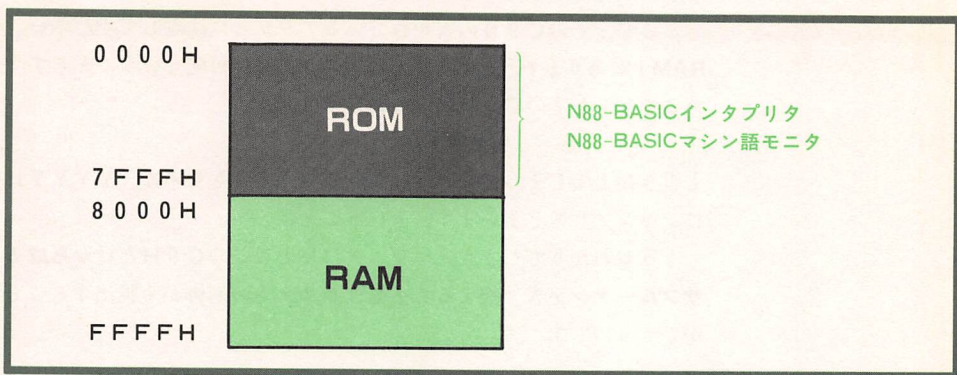
(利用不可能な機能を指定した)

が表示されました。ここらあたりがシャレたところです。とにかく、本来はROM内にあるはずの

MON――BASICのコマンド

が書き換えられてしまいました。

ROMとRAMのメモリ・マップ

ROM内のシステムでも書き換えが可能であることがわかりました。どのようにやったのか? それでは、その種明しとまいりましょう。

【図3・4】――PC-8801メモリ・マップ(モデル化)

これは、PC-8801のメモリ・マップです。これは、ROM部とRAM部の使用するアドレスをモデル化したものですから、精密なものではありません。ご存知のようにPC-8801のメモリ・マップは、メモリのバンク切り換えのため、もっと複雑です。この図で示したいことは、全アドレス空間を

前半(0000H～7FFFH):ROM

後半(8000H～FFFFH):RAM

のように使用している、ということです。

ところで、この図のとおりでしたら、ご覧のようにシステム部を書き換えることは不可能ですね?

空白のサブルーチン

あなたは、システム部を解析したことがありますか? あるとすれば、次のようなおかしな部分に出会いませんでしたか?

ある部分を解析していました。すると、

CALL　XXXXH

というのにぶつかりました。何かのサブルーチンだろうと思って、その番地を調べてみますと、

XXXXH＝C9H　―――RET

となっています。つまり、何もやらないサブルーチンをCALLしていたのです――、とマア、このような部分に。

フック・アドレス

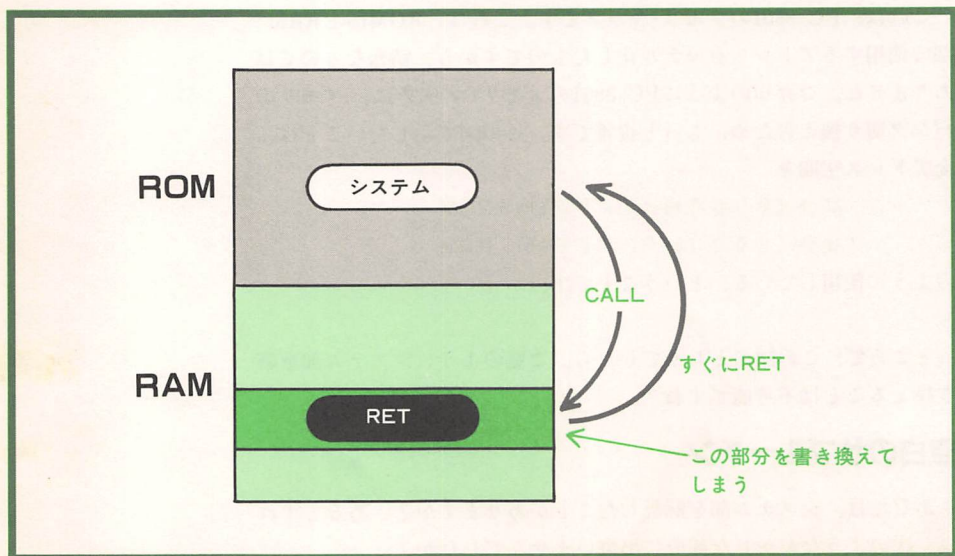
なんでこんなムダなことをしているのでしょう？　メモリは消費するし、実行時間も遅くなる……。

ここで、そのC9Hの書かれているアドレスに注意してください。RAM上にあります。ということは、書き換えが可能ということです。そこで、もしそのC9Hのところを

JP　YYYYH

と書き換えてしまったらどうなるでしょう？　もちろん、YYYYHにジャンプしてしまいます。ということは……。

もうおわかりでしょう。一見ムダに見えるこの**C9Hだけから成るサブルーチン**を書き換えることで、システムの制御から脱出することができるのです。

まとめますと、次のようになります。

【図3・5】――フック・アドレス

PC-8801のシステム部は、通常ROMの部分に書かれています。しかし、この図のように、ところどころRAMの部分に飛び出しているところがあります。ユーザーは、この部分を書き換えることでシステム部に変更を加えることができます。このようなRAM上に置かれた領域のアドレスを

フック・アドレス

と呼んでいます。

パーソナルMONコマンド

図3・3でMONが効かなくなったのも、図3・0のプログラムでMON処理ルーチンにおけるフック・アドレスを書き換えてしまったからです。そのしくみを次に説明いたします。

MON処理ルーチンの始めの部分に

```
E826H:CALL  69H
```

この部分はROMをCALLしている

というのがあります。そこで、69Hからのサブルーチンを調べてみますと、最初に

```
0069H:CALL  ED9FH
```

この部分はRAMをCALLしている

というのがあります。ここで**RAMの部分**をCALLしています。このED9FHが、フック・アドレスになります。この部分を書き換え、自分の都合の良いようにしてしまえば、自分専用のMONコマンドを作ることができます。いわば、**パーソナルMONコマンド**というわけです。

〈注〉ちなみに、ED9FHのサブルーチンは、

プロテクト・チェック：AD48H

をCALLしています。このルーチンは、現在ロードされているプログラムがPオプション付きかどうかチェックし、もしそうならプログラムにプロテクトをかけます。

プログラムの解析

以上が、図3・0のプログラムのしくみです。しくみを理解していただいた上で、最後にこのプログラムを解析しておきましょう。次にそのアセンブル・リストを示します。

```
                ;================================
                ;  KILL MON COMMAND:1983.6.6
                ;   ・ MESSAGE:1983.5.3
                ;================================
                ;
0038            MON:    EQU  38H                 ;MONITOR HOT START
5550            MSG:    EQU  5550H               ;PRINT MESSAGE
4DC1            ERR:    EQU  4DC1H               ;Feature not available
                ;
                        ORG  0BB00H
                ;
BB00 2107BB     SET     LD   HL,NOTMON       ┐
BB03 22A0ED             LD   (0EDA0H),HL     ├──────── ❶
BB06 FF                 RST  MON             ┘
                ;
BB07 2110BB     NOTMON:LD   HL,DMSG          ┐
BB0A CD5055             CALL MSG             ├──────── ❷
BB0D C3C14D             JP   ERR             ┘
                ;
BB10 0A2D2D2D   DMSG    DB   0AH,'------------------',0DH,0AH      ┐
BB14 2D2D2D2D                                                      │
BB18 2D2D2D2D                                                      │
BB1C 2D2D2D2D                                                      │
BB20 2D2D0D0A                                                      │
BB24 2020B1DD           DB   '  ｱﾝﾀ ﾅﾆ ﾔｯﾃﾝﾉ?',0DH,0AH              │
BB28 C020C5C6                                                      │
BB2C 20D4C2C3                                                      ├── ❸
BB30 DDC93F0D                                                      │
BB34 0A                                                            │
BB35 2D2D2D2D           DB   '------------------',0DH,0AH,0AH,00H   │
BB39 2D2D2D2D                                                      │
BB3D 2D2D2D2D                                                      │
BB41 2D2D2D2D                                                      │
BB45 2D0D0A0A                                                      │
BB49 00                                                            ┘
                ;
                        END
```

【図3・6】――アセンブル・リスト

❶ この部分が、フック・アドレスを書き換えている部分です。図3・1でこのプログラムを走らせています。これにより、

MON ⏎

とキーインすると、このプログラムが走るようになります。

❷ "アンタ　ナニ　ヤッテンノ？" を表示し、

Feauture not available

のエラー・ルーチンをCALLしている部分です。

❸ "アンタ　ナニ　ヤッテンノ？" のデータ・エリアです。

まとめ

以上のように、PC-8801では、ROM上にあるシステム・プログラムを修正、強化するため、

フック・エリア

が設けられています。ユーザーはこのフック・アドレスを書き換えることで、システムに変更を加えることができます。オリジナル・コマンドを作成することができます。あなたもシステムを解析していて、フック・アドレスを見つけたら、いろいろ実験を試みてください。だんだん、いろいろなことがわかってくるでしょう。

〈注〉PC-8001でも、フック・アドレスが設けられていました。たとえばN-BASIC ROMの中には、DISKとデータのやりとりを行うルーチンが用意されています。しかし、それは片面倍密用のサポートを行なっていました。そのルーチンの中にフック・アドレスをCALLしている部分があります。後に両面倍密用のDISKが現われた時、このフック・アドレスを利用して両面倍密用のサポート・ルーチンを作っていました。

3・1 マシン語モニタ・コマンドにおけるROMのセレクト法

ここで話が第0章の真中あたりにバックします。

我々は、最初〈MON処理ルーチン〉を解析しました。それによると、N-BASIC ROMには、

```
6000H=03H
6001H=CDH
```

が書き込まれている（はず）だということがわかりました。もしそうでないと、マシン語モニタが起動しないようになっていたのです。ところが実際にDコマンドで調べてみたら、そうはなっていませんでした（図0・7）。マシン語モニタが起動されたにもかかわらず——。

我々は、そろそろこの謎を解く時が来たようです。これにより、あなたはマシン語モニタにおける

ROMセレクトの強力な武器

を手に入れることになるでしょう。

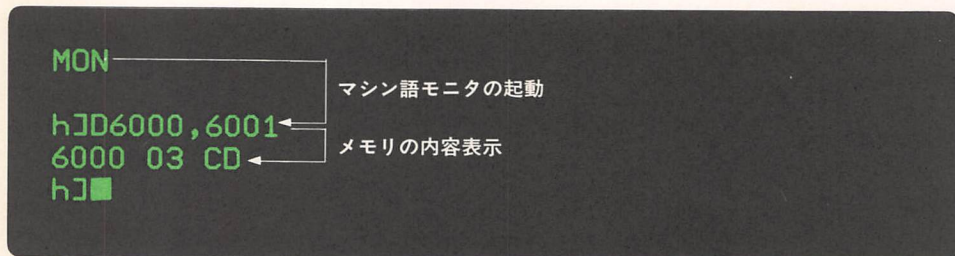
変わる6000H～6001H

まずは、次のとおりキーインを試みてください。

MON ⏎

でマシン語モニタを起動し、

D6000,6001 ⏎

でメモリの内容を調べます。

【図3・7】——ダンプ・メモリ

これは、図0・7と同じです。ところが本来は、次のようになっているはずです。

6000H	44H	N-BASIC ROM（この中にマシン語モニタが書かれている）のあるべき姿
6001H	42H	

ここでSコマンドを用いて

F1E1H=01H

のようにセットしてください。

```
h]SF1E1
F1E1 00-01
h]■
```

【図3・8】——F1E1Hを01Hに

そして、ふたたび

D6000,6001 ⏎

とキーインしてください。図0・5のようにN-BASIC ROMのあるべき姿が出現しました。

```
h]D6000,6001
6000 44 42
h]■
```

【図3・9】——N-BASIC ROMのあるべき姿

もう1つ。

Sコマンドで、

F1E1H=03H

のように変更してください。

```
h]SF1E1
F1E1 01-03
h]■
```

【図3・10】——F1E1Hを03Hに

そして、

D6000,6001 ⏎

です。

```
h]D6000,6001
6000 00 00
h]■
```

【図3・11】——新しい値の出現

ご覧のように、図3・7、図3・9とはまた違った新しい値が出現しました。これは、一体どうしたことでしょう？

F1E1Hをめぐって

どうやら

F1E1H

が曲者のようです。この値を変えることで、

6000H〜6001H

の値がいろいろに変わりました。そこで、F1E1Hについて探りを入れていくことに致します。

マシン語モニタ内で、F1E1Hを扱っているのは、

603DH————Ⓐ

722AH————Ⓑ

の二か所です。

まずⒶは、マシン語モニタの入り口、コールド・スタート部で、

```
XOR  A
LD   (F1E1H),A
```

のように、00Hをセットしています。したがって、MONでマシン語モニタが起動された直後は、必ず

F1E1H=00H

になっています。

次にⒷです。722AHを見ますと、

```
LD   A,(F1E1H)
```

となっていて、Aレジスタにこの値をセットしています。何かをやろうとしているのは、おわかりですね？

同一プログラムの出現

実は、この部分はあるサブルーチンの一部になっています。もし、その部分を見たいようでしたら、Sコマンドで

F1E1H=01H

にした後、

L7229,7253⏎

とキーインしてください。図3・12のように逆アセンブル・リストが得られます。722AHで、F1E1Hが使われているのがわかります。

次に、

LF18A,F1B4⏎

とキーインしてください。図3・13の逆アセンブル・リストが得られま

```
7229    F5          PUSH PSW
722A    3A F1E1     LDA  F1E1   ←F1E1Hの登場
722D    0F          RRC
722E    38 08       JRC  7238
7230    F1          POP  PSW
7231    CD F175     CALL F175
7234    7E          MOV  A,M
7235    C3 F16C     JMP  F16C
7238    0F          RRC
7239    38 03       JRC  723E
723B    F1          POP  PSW
723C    7E          MOV  A,M
723D    C9          RET
723E    F1          POP  PSW
723F    C5          PUSH B
7240    CD F175     CALL F175
7243    F3          DI
7244    DB 71       IN   71
7246    F5          PUSH PSW
7247    3E FE       MVI  A,FE
7249    D3 71       OUT  71
724B    46          MOV  B,M
724C    F1          POP  PSW
724D    D3 71       OUT  71
724F    78          MOV  A,B
7250    C1          POP  B
7251    C3 F16C     JMP  F16C
```

【図3・12】――逆アセンブル・リスト

す。そうしましたら、良く図3・12のリストと見比べてください。両者は、まったく同じであることがおわかりでしょう。

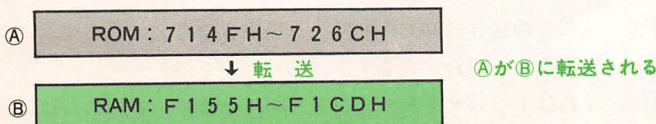
なぜ、同じプログラムが2か所にまたがって存在しているのでしょう。それは、第0章の図0・11❷を思い出していただければわかります。すなわち、マシン語モニタがコールド・スタートした時、

RAM上にモニタ用サブルーチンの一部を転送

する処理がおこなわれていましたね。この時

Ⓐ ROM：714FH～726CH

↓ 転　送　　Ⓐが Ⓑに転送される

Ⓑ RAM：F155H～F1CDH

```
F18A    F5          PUSH PSW  ←―――――サブルーチンの入口
F18B    3A F1E1     LDA  F1E1
F18E    0F          RRC
F18F    38 08       JRC  F199
F191    F1          POP  PSW  ┐
F192    CD F175     CALL F175 ├―F1E1H=00Hのときの処理
F195    7E          MOV  A,M  │
F196    C3 F16C     JMP  F16C ┘
F199    0F          RRC
F19A    38 03       JRC  F19F
F19C    F1          POP  PSW  ┐
F19D    7E          MOV  A,M  ├―F1E1H=01Hのときの処理
F19E    C9          RET       ┘
F19F    F1          POP  PSW  ┐
F1A0    C5          PUSH B    │
F1A1    CD F175     CALL F175 │
F1A4    F3          DI        │
F1A5    DB 71       IN   71   │
F1A7    F5          PUSH PSW  │
F1A8    3E FE       MVI  A,FE │
F1AA    D3 71       OUT  71   ├―F1E1H=03Hのときの処理
F1AC    46          MOV  B,M  │
F1AD    F1          POP  PSW  │
F1AE    D3 71       OUT  71   │
F1B0    78          MOV  A,B  │
F1B1    C1          POP  B    │
F1B2    C3 F16C     JMP  F16C ┘
```

【図3・13】――もう1つの逆アセンブル・リスト

のように転送が行なわれます。そして、図3・12を良くご覧になってください。

7229H～7253H

は、完全にⒶに含まれています。したがって、RAM上のⒷの領域に転送されます。それを再現したのが、図3・13です。これらの2つのプログラムがまったく同じなのは、きわめて当然です。

F18AHを解析する

さて、次にこの図3・13のプログラムの働きを考えてみます。これにより、大変なことがわかりますよ。

まず、これは1つのサブルーチンとなっています。その入口は、

F18AH

です。ですからこのサブルーチンを呼ぶには、

CALL F18AH

という形を取ります。

次に、このサブルーチンが、わざわざ

ROM———→RAM
転送

という手続きを経てCALLされていることに注意してください。ということは、このサブルーチンは、

メモリのバンク切り換えに関係がある

と予想されます。

そこで、このサブルーチンの概略です。

まず最初に、問題のF1E1Hの値を調べます。そして、その値により、次の3つの処理に分岐します。

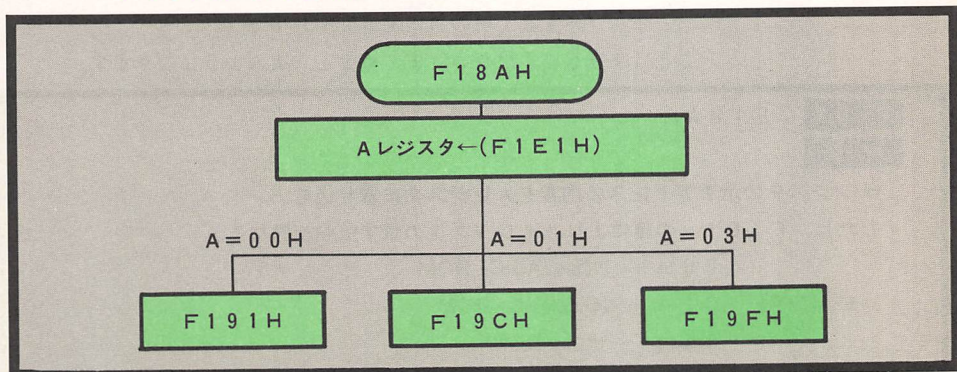
❶ **ビット0が0のとき（例　00H）**

F191H〜F198H

❷ **ビット0が1でかつビット1が0のとき（例　01H）**

F19CH〜F19EH

❸ **ビット0が1でかつビット1が1のとき（例　03H）**

F19FH〜F1B4H

【図3・14】——フローチャート

F1E1Hの機能

3つの処理のうち、もっとも簡単なのは

F1E1H=01H：F19CH〜F19EH

の場合です。リストを読めばすぐにわかりますが、

Aレジスタ←HLレジスタの示すアドレスの内容

の処理をしているだけです。

Ｆ１Ｅ１Ｈ＝００Ｈ：Ｆ１９１Ｈ～Ｆ１９８Ｈ

の場合は、

↓ N88-BASIC ROMに切り換える
Ａレジスタ←HLレジスタの示すアドレスの内容
N-BASIC ROMに戻す

の処理を行っています。

最後の、

Ｆ１Ｅ１Ｈ＝０３Ｈ：Ｆ１９ＦＨ～Ｆ１Ｂ４Ｈ

の場合は、

↓ N88-BASIC ROMに切り換える
６０００Ｈ～７ＦＦＦＨをサブROMに切り換える
Ｂレジスタ←HLレジスタの示すアドレスの内容
６０００Ｈ～７ＦＦＦＨを元のROMに切り換える
Ａレジスタ←Ｂレジスタ
N-BASIC ROMに戻す

の処理を行っています。

以上、いずれの場合においても結局は、

Ａレジスタ←HLレジスタの示すアドレスの内容

を行っていることになります。しかし、

Ｆ１Ｅ１Ｈの値によってROMが異なる

ことに注意してください。まとめると、次のようになります。

アドレス　Ｆ１８ＡＨ

機能

HLレジスタの示すアドレスの内容をＡレジスタに取り込む。

ただし、ＦＩＥＩＨの値により、HLレジスタの示すROMが異なる。

Ｆ１Ｅ１Ｈ { ００Ｈ――N88-BASIC ROM
０１Ｈ――N-BASIC ROM
０３Ｈ――サブ ROM

ROM別、Dコマンド利用法

それでは、このＦ１８ＡＨからのサブルーチンが、どんな場所で使われているかです。それは、たとえば、

Ｄコマンド処理ルーチン

の中で用いられています。HLレジスタが、現在処理中のアドレスを示しています。そして、

CALL Ｆ１８ＡＨ

で、Aレジスタにメモリの内容を取ってきます。そして、その値を表示します——ということをやっています。

ところで……。

ここからが重要です。

前にも述べましたように、マシン語モニタのコールド・スタート時に、

　　F1E1H＝00H

にセットされています。ということは、今調べたことから、通常

〈Dコマンド〉

0000H～7FFFH

については、N88-BASIC ROMの内容を表示する。

といえます。ところが、画期的なことは、

〈Dコマンド——その2〉

0000H～7FFFH

については、FIEIHの値を変更することで、任意のROMの内容を表示させることができる。

F1E1H＝
- 00H——N88-BASIC ROM
- 01H——N-BASIC ROM
- 03H——サブ ROM

(＊) 実際は、ビット0、ビット1の組み合わせがこうなっていれば、値は何でも可。

ということが判明いたしました。

〈注〉マニュアルには、このことが説明されていません。それどころか、Dコマンドで

0000H～7FFFH

を表示させる時、N88-BASIC ROMがセレクトされていることさえ言及されていません。

しかし、あなたのN88-BASICマシン語モニタには、この機能がついています。ただ、それがコマンド（あるいはパラメータ）としてスイッチする機能が省略されているだけです。自分で、FIEIHの値を変えてやれば、自由に、好きなROMの値を見るこができるわけです。このような技術情報も、自分でROMの中身を解析することで獲得できるのです。

以上のことから、

第０章：N-BASIC　ROMの謎

【図３・７】←→【図３・９】←→【図３・11】

の謎がすべて解けたました。これらは、すべて

ROMのバンク切り換え

のためで、それをあやつっていたのが、

Ｆ１Ｅ１Ｈ

だったのです。

本章の最後に、Ｆ１８ＡＨのCALLされているアドレスをすべて列挙しておきましょう。このことから、Ｆ１Ｅ１Ｈの値を変えることで、

Ｓ，Ｄ，Ｅ，Ｗ，Ｒ，Ｖ，Ｌ

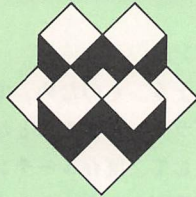
の各コマンドで、自由にROMをセレクトできることがわかります。

〈Ｆ１８ＡＨを参照しているアドレス〉

6157H	69E9H	6AA6H
6205H	69FDH	6AB0H
621BH	6A50H	6AB5H
67F4H	6A79H	
68D0H	6A89H	

第4章

ROM別システム・サブルーチンの利用法

4

PC-8801マシン語プログラマーへの技術情報、いよいよ大詰の最後の章となりました。マシン語のプログラミングは、一般的に手間のかかるものです。それは、データ移動を基本とする

マシン語の命令体系

のためでしょう。そのため、できるだけ効率良くプログラミングすることが必要になってきます。そして、既存ルーチンの再利用といったことも重要なことになります。その、もっとも合理的な手段が、

ROM内システム・サブルーチン

の利用ということでしょう。それは、マシン語プログラマーにとって、貴重な財産です。

ところが、PC-8801においては、このシステム・サブルーチンを使うのが、なかなかやっかいです。これは、同一アドレスに複数のROMを持つという、いわゆる

ROMのバンク切り換え

という問題があるからです。

本章では、この問題に決着をつけます。すなわち、

ROM別システム・サブルーチン
の利用法

への挑戦です。

また、本章の最後では、

CLEAR文の自動設定プログラム

をご紹介いたします。これによりマシン語の領域の自動確保がおこなわれます。本章、および前章までの技術情報を生かし、次章以下のマシン語プログラミングに挑戦して行ってください。

4·0　N88-BASIC ROM内システム・サブルーチン

PC-8801におけるマシン語の壁、それは

ROMのバンク切り換え

にあることは今まで見てきたとおりです。ROM内のシステム・サブルーチンでもこの問題をクリアするのに、複雑な方法を用いていました。まして、我々がユーザー・プログラムを作る際、この問題で悩まされるのは当然でしょう。マシン語プログラムがうまく動かない。暴走してしまう——といった時、この問題がからんでいることが多いようです。いかに〈ROMのバンク切り換え〉をフィニッシュするか——これは、PC-8801をターゲットとするマシン語プログラマーが避けて通ることのできない関所です。

N88-BASIC　ROM内システム・サブルーチン

PC-8801のユーザーが、ROM内のシステム・サブルーチンを利用する際、もっとも簡単なのは、

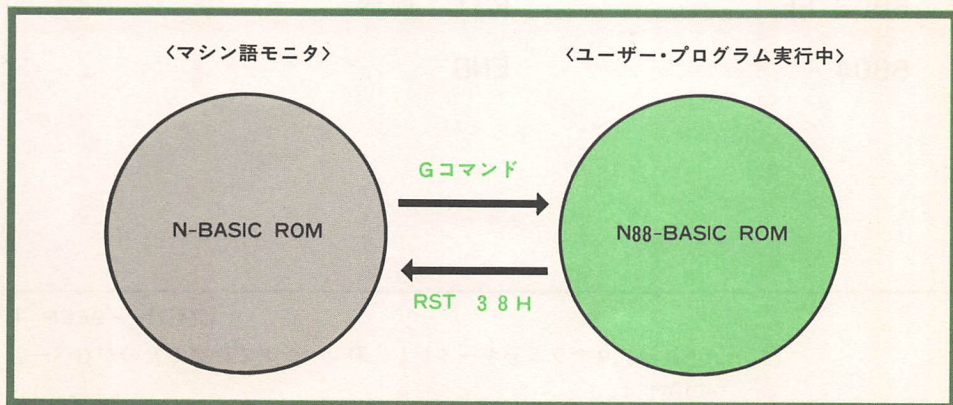
N88-BASIC　ROM

のシステム・サブルーチンを利用する方法です。なぜなら、すでに解析しましたように、

Gコマンド実行直後は、N88-BASIC　ROM
が選択されている

からです。ですから、N88-BASIC　ROM内のシステム・サブルーチンをCALLするのは、単純に

CALL 〈エントリー・アドレス〉

で実現できます。

【図4·0】——オリジナルなROMのセレクト

エレガントにCALL

実例でお目にかけましょう。N88-BASIC ROM内のシステム・サブルーチンを1つCALLしてみることに致します。次のサブルーチンは、いかがでしょう?

アドレス	3E9BH
機能	

一定時間、内蔵ブザーを鳴らす

このサブルーチンは、N88-BASIC ROMの中に書かれています。機能は、BASICの

BEEP 1

と同じです。単純にCALLすれば良いのですから、サンプル・プログラムは次のようになります。

```
                    ;===================
                    ;  BEEP:1983.6.20
                    ;===================
                    ;
0038                MON:    EQU  38H
3E9B                BEEP:   EQU  3E9BH
                    ;
                            ORG  0BB00H
                    ;
BB00 CD9B3E                 CALL BEEP            ;BEEP
BB03 FF                     RST  MON
                    ;
BB04                        END
```

【図4・1】――BEEP 1

このプログラムをキーインし、Dコマンドで確認したのが右ページの図です。

```
h]DBB00,BB03
BB00 CD 9B 3E FF
h]■
```

【図4・2】——Dコマンドで確認

OKでしたら、

GBB00 ⏎

でプログラムを走らせます。

ビー

と例のブザー音が聞こえ（←これは、エラーではありませんヨ）、次のようにプログラムが終了します。

```
h]GBB00  ——ブザーが鳴り、プログラムが終了する
h]■
```

【図4・3】——プログラム終了

以上により、次の〈ルール〉が確認されました。

> **〈ルール〉**
>
> N88-BASIC ROMに書かれたシステム・サブルーチンは
>
> CALL〈エントリー・アドレス〉
>
> で直接呼べばよい。

きわめて単純ですね？

4・1　N-BASIC ROM内システム・サブルーチン

前節に引き続き、今度は

N-BASIC　ROM

に書かれたサブルーチンをCALLすることを考えてみます。もともとマシン語モニタは、N-BASIC　ROM内に書かれているのですから、このROM内のシステム・サブルーチンをCALLするのは単純に見えます。しかし、実行してみますと、これがなかなか、どうして、どうして——。

マア、次の実例をご覧ください。

マシン語モニタ内システム・サブルーチン

ここで、CALLしてみようとするサブルーチンは、次の2つです。いずれも、N-BASIC　ROM内に書かれています。しかも、

6000H～7FFFH

にありますから、N88-マシン語モニタ内のシステム・サブルーチンです。

アドレス	7037H
機　能	
	0DH、0AHのコントロール・コードを出力する。

このサブルーチンの機能、平たく言えば

改　行　す　る

ということです。カーソルが、次の行の先頭に移ります。

アドレス	6F9AH
機　能	
	HLレジスタの値を16進数4桁で表示する。

マシン語モニタ内で、さかんに使われます。非常に便利なサブルーチンで、アドレスの表示や、レジスタ・ペアの表示などに使えます。自分でモニタなどを作る時に利用するとよいでしょう。

N-BASIC　ROMにスイッチ

これらのサブルーチンをCALLするには、単純に

CALL　〈エントリー・アドレス〉

のように呼ぶことは、できません。なぜなら、

G××××　⏎

でプログラムを走らせた直後、ROMが

N88-BASIC ROM

に切り換わっているからです。

「それなら、一度、N-BASIC ROMに切り換えてから、サブルーチンをCALLすればいいじゃないか」

となります。まさにその通りで、次のシステム・サブルーチンをコールすることで、N-BASIC ROMをセレクトすることができます。

アドレス F16CH

機能

N-BASIC ROMをセレクトする。

このサブルーチンは、RAM上にありますかから、前に説明しましたように、

どのROM上からもCALL可能

となっています。

以上を用いて、サンプル・プログラムを作ると次のようになります。これで良いでしょうか?

```
                    ;=========================
                    ;  PRINT HL-0:1983.6.20
                    ;=========================
                    ;
  0038              MON:   EQU  38H
  6F9A              PHL:   EQU  6F9AH
  7037              CRLF:  EQU  7037H
  F16C              XNBAS: EQU  0F16CH
                    ;
                           ORG  0BB00H
                    ;
❶ BB00 CD6CF1              CALL XNBAS            ;SET N-BASIC ROM
❷ BB03 CD3770              CALL CRLF             ;LINE FEED
❸ BB06 213412              LD   HL,1234H
❹ BB09 CD9A6F              CALL PHL
❺ BB0C FF                  RST  MON
                    ;
                           END
```

【図4・4】――サンプル・プログラム

ROMの落とし穴

これで良いでしょうか?――とは、いやなことを聞きますね。マア、最初にプログラムの意味を考えてみましょう。図の番号にしたがって説明して行きます。

❶　まずは、N-BASIC　ROMに切り換えます。このROMに書かれているシステム・サブルーチンをCALLするためです。

❷　N-BASIC　ROM内システム・サブルーチンその１のコールです。１行改行します。

❸　HLレジスタに１２３４Ｈをセットします。次のサブルーチンでこの値を表示します。

❹　N-BASIC　ROM内システム・サブルーチンその２のコールです。このサブルーチンにより、❸でセットしたHLレジスタの値がＴＶ画面に出力されます。

❺　処理がすべて終了しましたので、リスタート命令によりブレークをかけ、マシン語モニタに戻ります。

ザッと見ますと、このようになっています。これで良さそうですね。おっと！　ここに落とし穴があるのです。

❶～❹は、これでＯＫです。ここまでは、正しく動きます。

問題は、❺にあります。RST　３８Ｈでマシン語に戻しているのだから、これでよさそうにみえます。しかし、あのブレーク・ポイント設定のしくみを解析した時のことを良く思い出してください。

RST　３８Ｈ　→３８ＨをCALL

でした。３８Ｈには、何が書かれていたか？　2・2でやりましたように、

```
JP      E669H
        └──────このアドレスに注意
```

となっていました。ところで、この命令はどこに書かれていましたか？

――N88-BASIC　ROM！

そうです。RST　３８Ｈでブレークをかけ、マシン語モニタをホット・スタートさせるには、

N88-BASIC　ROMがセレクトされていること

が必要なのです。ヨカ、ヨカ？　N-BASIC　ROMがセレクトされている時に、RST　３８Ｈを実行しても、たしかに３８Ｈにジャンプします。しかし、そこには、

```
JP      F1E3H
        └──────N88-BASIC　ROMとは異なる
```

と書かれています。これでは、マシン語モニタには戻れません。ここのところ、おわかりいただけたでしょうか？

ファイル名にジャンプ

ウソかどうかは、実行してみればわかります。図4・4のプログラムを入力してみましょう。

```
h]BB00,BB0C
BB00 CD 6C F1 CD 37 70 21 34 12 CD 9A 6F FF
h]■
```

【図4・5】――ダンプ・リスト

そして、

GBB00 ⏎

で実行です。

【図4・6】――暴走!

ご覧のように、HLレジスタの値が表示されるところまでは順調に動きます。しかし、その後がいけません。暴走です。

どんな暴走が起こるか? それは不定です。と申しますのは、N-BASIC ROMがセレクトされていて38Hに飛んだ時、

F1E1H

にジャンプします。しかし、そこは〈ファイル名〉のワーク・エリアとなっており、何が書かれているかは不定だからです。

N-BASIC ROM内ルーチンの正しい利用法

どうやら、N-BASIC ROM内システム・サブルーチンを呼ぶ〈正しい方法〉がわかってきました。それは次のようになります。

最後に必ず

N88-BASIC ROM に戻す!

というのが重要です。その方法は、次のシステム・サブルーチンを利用すればOKです。これもRAM上にありますから、どのモードからでも使用できます。ご愛用を。

アドレス	F175H
機 能	

N88-BASIC ROMに切り換える

プログラムの修正

以上の知識をもとに、図4・4のプログラムを正しく変更してみましょう。

```
                    ;======================
                    ;  PRINT HL:1983.6.20
                    ;======================
                    ;
0038                MON:    EQU  38H
6F9A                PHL:    EQU  6F9AH
7037                CRLF:   EQU  7037H
F175                XN88:   EQU  0F175H
F16C                XNBAS:  EQU  0F16CH
                    ;
                            ORG  0BB00H
                    ;
BB00 CD6CF1                 CALL XNBAS                ;SET N-BASIC ROM
BB03 CD3770                 CALL CRLF                 ;LINE FEED
BB06 213412                 LD   HL,1234H
BB09 CD9A6F                 CALL PHL
BB0C CD75F1                 CALL XN88                 ;SET N88-BASIC ROM
BB0F FF                     RST  MON
                    ;
                            END
```

追 加

【図4・7】——修正プログラム

ご覧のように、

CALL F175H

を追加してあります。入力して、確認します。

```
h]DBB00,BB0F
BB00 CD 6C F1 CD 37 70 21 34 12 CD 9A 6F CD 75 F1 FF
h]■
```

【図4・8】——Dコマンドで確認

そして、

GBB00 ⏎

で実行です。

```
h]GBB00
1234
h]■
```

【図4・9】――実行

ご覧のように、見事（？）にマシン語モニタに戻りました。どうもオメデトウございます。

〈注〉図4・7のプログラムにおいて、BB03Hの

CALL 7037H

の働きがわかりにくいかもしれません。この行は、特に必要というわけではありません。ためしに取り除いて実行してみると、その働きがつかめるでしょう。ここでは、実行後、コマンド・ラインが消去されてしまうとだけ申し添えておきます。

XコマンドをCALL

N-BASIC ROM内システム・サブルーチンの利用、最後にもう1つだけ実例をお目にかけましょう。次のようなマシン語モニタ内のサブルーチンは、いかがでしょう？

アドレス	62DAH
機能	スタックされている全レジスタの値を表示する

これは、モニタのXコマンドで全レジスタを表示する場合のサブルーチンです。セオリ通りに

N-BASIC ROMをセレクトする
↓
CALL 62DAH
↓
N88-BASIC ROMに戻す

とプログラムをしたのが、次の図4・10です。このマシン・コードを入力し、

GBB00 ⏎

で走らせたのが、その次の図4・11です。ご覧のようにいつものXコマンドの画面が得られました。これは、これでもちろんうまく行っています。しかし、――。

```
                    ;===================
                    ;  X-ALL:1983.6.20
                    ;===================
                    ;
0038                MON:   EQU  38H
62DA                XALL:  EQU  62DAH
F175                XN88:  EQU  0F175H
F16C                XNBAS: EQU  0F16CH
                    ;
                           ORG  0BB00H
                    ;
BB00 CD6CF1                CALL XNBAS              ;SET N-BASIC ROM
BB03 CDDA62                CALL XALL
BB06 CD75F1                CALL XN88               ;SET N88-BASIC ROM
BB09 FF                    RST  MON
                    ;
                           END
```

【図4・10】――アセンブル・リスト

```
h]GBB00
A :00 F :PZ---E-- B :0000 D :EDCC H :0001 A´:00 F´:PZ---E-- B´:0000 D´:BB0A
H´:BE0F IX:CEF1 IY:D026 I :F3 PC:0000 SP:B8FB
h]■
```

【図4・11】――レジスタ類が表示されて

実は、このプログラムにおいては、ＢＢ０６Ｈの

CALL　F１７５H

は不要です。それどころか、

RST　３８H

もいりません。

ナゼか？

６２ＤＡＨをCALLして、レジスタ類を表示した後、自動的にマシン語モニタに戻ってしまうからです。世の中、なかなかセオリ通りには行かないもので、マシン語の世界でもことさらそのようです。ただ今のは、その見本のようなものです。最後はご愛敬ということでこの節、おしまい。

4・2　サブROM内システム・サブルーチン

ROM別システム・サブルーチンの利用法、最後は

サブROM（グラフィックROM）

の利用です。ご存知のようにPC-8801ではグラフィック関係のサブルーチンを第3のROMに収め、一般に**サブROM**と呼んでいます。ROMのアドレスは、

6000H～7FFFH

です。N88-BASIC　ROMの一部をサブROMに切り換えるには、N88 BASICモードで

```
LD    A,FEH    ┐
OUT   (71H),A  ┘───Ⓐ
```

を実行します。また、もとに戻すには、

```
LD    A,FFH    ┐
OUT   (71H),A  ┘───Ⓑ
```

とします――と一般に言われています。サブROMをCALLするのに、いちいちこんなことをしなければならないのでしょうか？　これでは、システム・サブルーチンを利用する意味がありません。

N88-BASICは、たくさんのグラフィック命令をサポートしています。そして、そのたびにⒶⒷのようなことをしているのでしょうか？おそらく、サブROMを利用するためのうまいルーチンがあるはずです。ここらあたりの秘密を探っていくのが、本節のねらいです。

CD　51　45　××

最初にお断りしておきますが、この章は少々難しいかもしれません。ですから、初めはサラリと流し読みをし、

サブROMの利用法

だけを理解すればよいと思います。マア、マシン語でグラフィック命令を制御したい方は、がんばってマスターしてください。

そこで本論です。

まず、

D6E96, 6F11 ⏎

とキーインしてみてください。面白いデータが得られます。それが、次の図4・12です。

このデータをよくご覧になってください。

CD　51　45　××

というデータが並んでいるのがわかります。××の部分は

00 01 ～ 1E

となっているのもわかります。これは、何かありそうですね。

```
h]D6E96,6F11
6E96 CD 51 45 00 CD 51 45 01 CD 51 45 02 CD 51 45 03
6EA6 CD 51 45 04 CD 51 45 05 CD 51 45 06 CD 51 45 07
6EB6 CD 51 45 08 CD 51 45 09 CD 51 45 0A CD 51 45 0B
6EC6 CD 51 45 0C CD 51 45 0D CD 51 45 0E CD 51 45 0F
6ED6 CD 51 45 10 CD 51 45 11 CD 51 45 12 CD 51 45 13
6EE6 CD 51 45 14 CD 51 45 15 CD 51 45 16 CD 51 45 17
6EF6 CD 51 45 18 CD 51 45 19 CD 51 45 1A CD 51 45 1B
6F06 CD 51 45 1C CD 51 45 1D CD 51 45 1E
h]■
```

【図4・12】――画白いデータ列

謎の逆アセンブル・リスト

これらのデータ列（←マシン語かもしれない）は、

CD 51 45 ××

という4組のデータから構成されています。ためしに、この部分を逆アセンブルしてみると、次のようになります。

〈アドレス〉	〈マシン・コード〉	〈アセンブリ言語〉
6E96H	CD 51 45	CALL 4551H
6E99H	00	DB 00H
6E9AH	CD 51 45	CALL 4551H
6E9DH	01	DB 01H
6E9EH	CD 51 45	CALL 4551H
6EA1H	02	DB 02H
	∫	
6E0EH	CD 51 45	CALL 4551H
6F11H	1E	DB 1E

【図4・13】――データ列の逆アセンブル

以上のように31組の

```
CALL 4551H
DB   nn     (nn:00H～1EH)
```

が得られました。ここで、次のような疑問が生まれます。

●この逆アセンブルは正しいものか？

●もし正しいとしたら、

マシン語の命令（CALL）

データ

が交互に現われており、このままでは暴走してしまうのでは？

解析の糸口

この疑問は、もっともです。そこで、それに対する解答です。

——答。正解です。これらは、グラフィックを処理するため、サブROMをCALLするために設けられています。サッと見ると、

```
CALL  4551H
DB    nn
```

で、無茶苦茶なデータ列に見えます。しかし、このデータがあるためメインのN88-BASIC ROMに収められているBASICインタプリタから、容易にサブROM内グラフィック処理ルーチンを利用することができるのです。

どのようにして？ それは、

CALL 4551H ―― このサブルーチンを解析

することによって解明いたします。こうして我々の次の目標が設定されました。

4551Hのサブルーチンを解析する

——これが、次の目標です。

サブROMコール・ルーチンの解析

ところが、この部分、比較的解析しにくくなっています。と申しますのは、この部分に、

巧妙なスタック操作

が用いられているからです。このインタプリタを制作した人は、スタック操作を楽しんでいるのではないかと思われるくらいです。そのへんを覚悟の上で、次の解析に取り組んでください。

逆アセンブル・リストを次の図4・14に示します。例によって説明は図の番号にしたがっておこないます。

❶ ROMのバンクを切り換えるため、その間に割込みが入るのを禁止します。

```
❶4551   F3        DI
❷4552   22 EF0B   SHLD EF0B  ←―― HLをストア
 4555   CD EDF3   CALL EDF3  ←―― フック・アドレス
 4558   32 EF0A   STA  EF0A  ←―― Aをストア
❸455B   DB 71     IN   71    ┐
 455D   E1        POP  H     │ HL←DBのアドレス
 455E   F5        PUSH PSW   ┘ I/O71Hのデータをスタック
❹455F   3A EF0A   LDA  EF0A  ←―― Aをロード
❺4562   E5        PUSH H     ┐ RET→4581Hへ
 4563   21 4581   LXI  H,4581│ HL←DBのアドレス
 4566   E3        XTHL       ┘
❻4567   D5        PUSH D
 4568   F5        PUSH PSW
❼4569   6E        MOV  L,M   ┐
 456A   26 00     MVI  H,00  │ HL←サブルーチンNo.X2
 456C   29        DAD  H     ┘
❽456D   3E FE     MVI  A,FE  ┐
 456F   D3 71     OUT  71    ┘ サブROMセレクト
❾4571   11 600D   LXI  D,600D┐
 4574   19        DAD  D     ┘ HL←ジャンプ・アドレス
❿4575   5E        MOV  E,M   ┐
 4576   23        INX  H     │
 4577   56        MOV  D,M   │ HL←ジャンプ・アドレス
 4578   EB        XCHG       ┘
⓫4579   F1        POP  PSW
 457A   D1        POP  D
⓬457B   E5        PUSH H     ←―― RET→ジャンプ・アドレス
⓭457C   2A EF0B   LHLD EF0B  ←―― HLをロード
⓮457F   FB        EI
 4580   C9        RET        ←―― サブルーチンをCALL
⓯4581   F3        DI         ←―― サブルーチンよりリターン
⓰4582   32 EF0A   STA  EF0A  ←―― Aをセーブ
⓱4585   E3        XTHL       ┐
 4586   7C        MOV  A,H   │ N88-BASIC ROMへ
 4587   D3 71     OUT  71    ┘
⓲4589   E1        POP  H     ┐
 458A   3A EF0A   LDA  EF0A  ┘ レジスタ復活
⓳458D   FB        EI
 458E   C9        RET
```

【図4・14】――サブROMコール・ルーチン

❷ Aレジスタ、HLレジスタの値をサブROM内ルーチンに運ぶため、ワーク・エリアにストアします。

❸　HLレジスタに

```
XXXXH   nn        DB   nnH
```
（XXXXH：このアドレスをHLレジスタにセット）

のアドレスをセットします。さらに、N88-BASIC　ROMに戻すための出力データをスタック領域にストアします。

❹　Aレジスタの値を復活します。

❺　サブROM内サブルーチンからリターンしてきたとき、⓯の

4581H

へジャンプするようにスタック領域の値を調整します。

❻　Aレジスタ、DEレジスタの値をスタック領域に保存します。

❼　ここは、面白いことをしています。

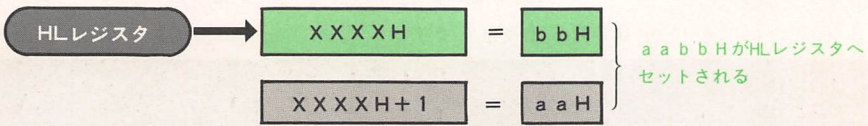
DB　nnH

で定義された値をHLレジスタにセットし、かつその値を2倍しています。

❽　6000H～7FFFHをサブROMに切り換えます。

❾　ここで、また面白いことをやっています。❼でセットされたHLレジスタの値に、

```
600DH
```
（600DH：この場合、❽によりサブROM内のアドレスになっている）

を加えています。すなわち、これ以後HLレジスタはサブROM内の600DH以降のあるアドレスをポイントすることになります。一体、サブROM内600DH以降には何が書かれているのでしょう。

というようなことをやっています。

⓫　❻の値を復活します。

⓬　RET命令を発した時に、

aabbH

にジャンプするようになります。

⓭　HLレジスタの値を復活します。

⓮　割込みを許可し、

aabbH

にジャンプします（実は、これがサブROM内グラフィック処理ルー

チンをCALLしていることになります)。

⑮ サブROM内のルーチンからリターンして来ると、ここに飛び込みます。ROMを元に戻すため、割り込みの禁止をします。

⑯ サブルーチンから戻ってきた時のAレジスタの値をワーク・エリアにストアします。

⑰ 6000H〜7FFFH

をメインのN88-BASIC ROMに戻します。

⑱ レジスタ類を復活します。

⑲ 割り込みを許可します。そして、大変な事が起こります。何と、❶をCALLした時の、

1つ浅いスタックのアドレスへリターン

するのです。

サブROM内ジャンプ・テーブル

以上、スタック関係がややっこしく、理解しづらかったことと思います。しかし、先に進みましょう。

まず、❾で出てきた

600DH〜

にどんなデータが入っているかを見てみます。ここには、

DB nn (nn:00H〜1EH)

に対応する31組の各処理に対するジャンプ・テーブルが書き込まれています。そのジャンプ・テーブルは、次のようにして見ることができます。

最初に、サブROMを見るため3・1でやりました

ROMセレクト・スイッチ=F1E1H

を03Hにセットします。

```
h]SF1E1
F1E1 00-03
h]■
```

スイッチをサブROMにセット

【図4・15】——サブROMにセット

ジャンプ・テーブルは2バイトずつ31組ありますから、

D600D, 604A ↵

とキーインしてください。次のようにジャンプ・アドレスが表示されます。

このテーブルの見方は、次のようになっています。

たとえば、

```
hJD600D,604A
600D FB 7D 00 7E 29 6D 33 7C 53 70 64 70 8B 7E 25 6E
601D 00 67 C6 6A 19 7E C0 6D 78 68 BC EE D6 79 D5 6C
602D 55 6C 74 76 8F 69 A8 6C 94 6A 0A 6D D0 72 ED 72
603D 24 73 2E 74 2A 75 EE 74 A1 75 4F 75 DD 75
hJ■
```

００に対応するジャンプ・アドレス

１Ｅに対応するジャンプ・アドレス

【図4・16】――ジャンプ・テーブルを表示する

DB　　1EH

に対応するジャンプ・アドレスは、１番最後の

６０４９Ｈ＝ＤＤＨ ６０４ＡＨ＝７５Ｈ	→７５ＤＤＨ

により、７５ＤＤＨとなります。

サブルーチン番号

ここまで解明できれば、サブROMの使い方が具体化します。次の使い方をよくマスターしてください。

サブROM内サブルーチンで、

CALL　４５５１Ｈ

でCALLできるのは、31組あります。そして、それぞれのサブルーチンに、

００Ｈ～１ＥＨ

の番号がついています。仮に、この番号を

サブルーチン番号

と呼ぶことにします。各サブルーチンは、たとえば次のように〈サブルーチン番号〉で区別できます。

〈サブルーチン番号〉

００Ｈ――PRESET処理
０１Ｈ――PSET処理
～
１ＥＨ――RENUM処理

2つのDATA列

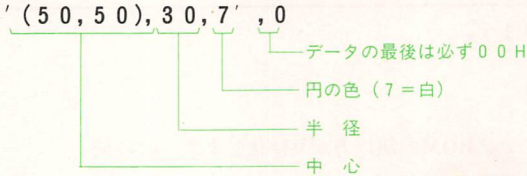
次に実際にサブROM内サブルーチンをCALLしてみます。例として、次のサブルーチンを呼んでみましょう。

〈サブルーチン番号〉　14H
〈　処　　理　〉　CIRCLE

これは、N88-BASICの

CIRCLE処理――円を描く

を行うサブルーチンです。

それでは、CALLしてみます。まず、メモリ上の適当な領域に

```
CALL 4551H
DB   14H
```

という4バイトのデータを用意します。それと、もう1つ。

CIRCLE文のパラメータ

をキャラクター・コードで表わしたデータを用意します。たとえば、次のように。

```
DB  '(50,50),30,7',0
```

- 0 ── データの最後は必ず00H
- 7 ── 円の色（7＝白）
- 30 ── 半径
- (50,50) ── 中心

以上、2つのデータを用意しましたら、次のようにしてサブルーチンをCALLします。

```
LD     HL, <パラメータのアドレス>
CALL   <4バイトのデータのアドレス>
```

マシン語で円を

以上の原理のもとに、これをプログラム化したのが、次の図4・17です。

説明致します。最初の

CALL　5F0EH

は、サブROMには関係ありません。N88-BASIC ROM内のサブルーチンです。

アドレス　5F0EH

機　能

テキスト画面をクリアする。

CIRCLE文のパラメータ用DATAを準備しているのが、

BB0EH

```
                  ;====================
                  ;  CIRCLE:1983.6.20
                  ;====================
                  ;
0038              MON:    EQU  38H
4551              ROM5:   EQU  4551H
5F0E              CLS:    EQU  5F0EH
F175              N88:    EQU  0F175H
                  ;
                          ORG  0BB00H
                  ;
BB00 CD0E5F       MAIN:   CALL CLS
BB03 210EBB               LD   HL,DATA
BB06 CD0ABB               CALL SUB                    ;PRINT CIRCLE
BB09 FF                   RST  MON
                  ;
BB0A CD5145       SUB:    CALL ROM5
BB0D 0E                   DB   14                     ;CIRCLE
                  ;
BB0E 2835302C DATA:       DB   '(50,50),30,7',0
BB12 3530292C
BB16 33302C37
BB1A 00
                  ;
                          END
```

【図4・17】――CIRCLE処理ルーチンのCALL

です。

中心――(50, 50)

半径―― 30

色 ―― 白

にセットしています。また、ＢＢ０ＡＨで４バイトのDATAを準備しています。そして、ＢＢ０３Ｈで

HL←パラメータの先頭アドレス

をセットした後、ＢＢ０６Ｈで

CALL 〈４バイトのDATA〉

を実行して、サブROM内の

CIRCLE処理ルーチン

をCALLしています。

ＢＢ０９Ｈが、プログラム・エンドです。

それでは、プログラムを入力し、実行してみます。図４・18が

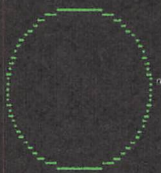
ＤＢＢ００，ＢＢ１Ａ ⏎

でプログラムの確認をしているところです。そして、

ＧＢＢ００ ⏎

でプログラム・スタートです。

```
hDBB00,BB1A
BB00 CD 0E 5F 21 0E BB CD 0A BB FF CD 51 45 0E 28 35
BB10 30 2C 35 30 29 2C 33 30 2C 37 00
h]■
```

【図4・18】——ダンプ・リスト

【図4・19】——円が表示される

ご覧のように、円が表示されました。うまく、サブROMが働いたのです。マシン語でCIRCLE文をあやつることに成功しました。

ROM内DATA列の利用

サブROMの利用法、おわかりいただけたでしょうか？

さて、賢いあなたは図4・17のプログラムをご覧になって、

「ずい分ムダなことをしている！」

と思ったのではないでしょうか？

ＢＢ０ＡＨをご覧ください。

```
CALL  4551H
DB    14H
```

となっています。しかし、このDATAでしたらユーザーがわざわざ用意しなくても、ROM内にあったではありませんか？　すなわち、図4・12の中に31組分、すべて用意されています。

6E96H〜6F11H

のDATA列は、まさにこのために用意されているのでした。

このDATA列を利用して、図4・17を書き直したのが、次の図4・20です。そして、そのダンプ・リストが図4・21です。もちろん、プログラムは正しく動きます［図4・22］。

```
                ;======================
                ;  CIRCLE-2:1983.6.20
                ;======================
                ;
0038            MON:    EQU  38H
6ECE            CIRCLE:EQU  6ECEH
5F0E            CLS:    EQU  5F0EH
                ;
                        ORG  0BB00H
                ;
BB00 CD0E5F     MAIN:   CALL CLS
BB03 210ABB             LD   HL,DATA
BB06 CDCE6E             CALL CIRCLE              ;PRINT CIRCLE
BB09 FF                 RST  MON
                ;
BB0A 2835302C DATA:     DB   ´(50,50),30,7´,0
BB0E 3530292C
BB12 33302C37
BB16 00
                ;
                        END
```

【図4·20】——DATA列を利用して

```
h]DBB00,BB16
BB00 CD 0E 5F 21 0A BB CD CE 6E FF 28 35 30 2C 35 30
BB10 29 2C 33 30 2C 37 00
h]■
```

【図4·21】——ダンプ・リスト

【図4·22】——実行

4・3　CLEAR文自動設定プログラム

さあ、第4章もいよいよ終りに近づいてきました。そして、

マシン語プログラマーへの技術情報

も終りに近づいたことになります。というよりは、前節までで必要なマシン語技術情報は、大体出揃ったと言えましょう。ここまでマスターされたあなたは、PC-8801でマシン語を使いこなすのに十分なパスポートを手に入れたことになります。あとは必要なI／Oデバイスを制御するだけです。

それについては、

ROM内システム・サブルーチン

がほとんどサポートしてくれるでしょう。そのヒントは、次章以下で提供されるはずです。そして、それらのルーチンを利用するのに十分な知識を、すでにあはたは獲得しているわけです。

あと1つ、残る問題——。

それはたった1つ。マシン語プログラムを、

メモリ中のどの領域に置くか

という問題です。いや、マシン語のプログラムは、どの領域まで使えるかという〈境界線〉の問題が残るだけです。

これについては、何ら考える必要はありません。それは、自動的に答が出る問題です。

どのようにして？

マア、以下に掲げるプログラムの使い方をマスターしてください。

マシン語プログラミング領域

マシン語プログラムのユーザー領域

——これは、重要な問題にかかわらず、従来、明確な資料がありませんでした。その中でも、最も詳しい資料、——それは皮肉にも

「PC-8801　USER'S　MANUAL」

です。このマニュアルのP.15-2が貴重な資料となっています。それによりますと、

XXXXH～E5FFH

が、ユーザーの使用できるマシン語領域です。そして、そのためには

CLEAR　,　XXXXH－1 ⏎

をあらかじめ実行しておく必要があります。

逆に言えば、

CLEAR　,　ZZZZH　⏎

を実行しておけば、

```
ZZZZH+1 ～ E5FFH
               └────────この部分は固定
    └──────────────────この部分は不定
```

の間で、自由にマシン語が使えるわけです。

この境界の後端、Ｅ５ＦＦＨは一定です。これより後ろにずらすことはできません。しかし、境界の前端ＺＺＺＺは可変です。この値を小さくすればするほど、

マシン語プログラミング可能領域が拡大

するわけです。

それでは、ＺＺＺＺＨはどこまで下げることが可能でしょうか？

マシン語プログラミング領域の自動計算

これについてマニュアルは、

「(CLEAR文が) エラーにならずに実行できたならば、

[a～E5FF]

の間で機械語プログラムを作っても、BASICが壊すことはありません」

といった、きわめてあいまいな表現を使っています。つまり、

「適当にCLEAR文を使ってみてください。運が良ければそこからマシン語が使えますよ」

といった表現をしているのです。したがって、この問題については、製品のマニュアルをはじめとして、今まで明確な資料が存在していませんでした。これでは、マシン語プログラマーは自らの勘に頼るしかありません。

さあ、そこで次のプログラムをご覧ください。このプログラムが、自動的にこの問題の解答を与えてくれます。

ユーザーが使えるマシン語プログラミング領域——。

それは、ディスクを使うか？　使うとすればディスク・ユニットを何台くらい使うか？　ファイル・バッファはいくつ位使うか——などにより違ってきます。次のプログラムは、そのおのおのの状態について、必要な領域を自動的に計算してくれるものです。

CLEAR文自動設定プログラム

それでは、図４・23のプログラムを実際に使ってみましょう。ファイル・バッファの数などを適当に定めてください。そして、このプロ

```
1000 '-----------------------------------
1010 '   SET CLEAR FOR MACHINE LANGUAGE
1020 '       1983.6.20:K. TSUKAGOSHI
1030 '-----------------------------------
1040 '
1050 LABLTOP=&HEB16
1060 MAX=PEEK(LABLTOP)+PEEK(LABLTOP+1)*256+&H215
1070 PRINT "SET ";:COLOR 6:PRINT "CLEAR ,&H";HEX$(MAX);
1080 COLOR 0:PRINT " ('Y' or 'N')";
1090 I$=INPUT$(1)
1100   IF I$="N" OR I$="n" THEN END
1110   IF I$<>"Y" AND I$<>"y" THEN 1100
1120 CLEAR ,MAX:PRINT :PRINT "▬▬  Complete !  ▬▬"
```

【図4·23】──CLEAR文自動計算プログラム

グラムを入力してください。そうしましたら、

RUN ⏎

でプログラム・スタートです。次のように表示されるでしょう。

```
RUN
SET CLEAR ,&HB294 ('Y' or 'N')■
```

入力を促すカーソル

CLEAR文の設定値が自動計算される

【図4·24】──実行開始

ご覧のように、CLEAR文で必要なパラメータの値が自動的に計算されて表示されます。そして、

Y または N

のいずれかのキーを押すように要求されます。キーの選び方は、次のとおりです。

Y ──表示されたCLEAR文が実行される

N ──表示されただけでCLEAR文は実行されない

N の場合は、CLEAR文は実行されませんが、表示された値をメモしておくことで、後でその値を利用することができます。ここでは Y を押してみます。

```
RUN
SET CLEAR ,&HB294 ('Y' or 'N')
▬▬  Complete !  ▬▬
Ok
■
```

【図4·25】──Y を実行

ご覧のように

Complete！

が表示され、プログラムの実行が停止します。これで

CLEAR ，&HB294

が実行されたことになります。以後、

B295H～E5FFH

の間でマシン語のプログラムを使うことができます。

〈注〉図4・24で表示される値は、その時の

接続されているディスク・ユニットの数

ファイル・バッファの数

により異なります。

マシン語活用最大領域

ここで、もう一度図4・23のプログラムを走らせてみます。

RUN ⏎

すると、次のように、

Out of memoryエラー

が発生します。

【図4・26】――Out of memoryエラー

一度走ったプログラムが、二度目にはエラーとなりました。なぜでしょう？

おわかりですね？　図4・23のプログラムは、

BASICの使用領域を最小限に制限

するように設定されます。したがって、

マシン語領域は最大限に確保

されます。このため、BASICの変数などを使用するための領域がなく

なってしまったのです。

ですから結論としては、次のように言うことができます。

BASICは一切使わず、マシン語だけを使う方は、図4・24で表示された値でCLEAR文を設定してください。目いっぱいマシン語領域が確保されます。しかし、BASICも併用したい方は、

```
CLEAR ,&HB294+2222
```

2222 ── BASICで使用するメモリ数

のように設定し、マシン語領域の一部をBASICに明け渡してやってください。

CLEAR文の限界

最後に、図2・24で表示された値がマシン語活用最大領域であるのを確認してみましょう。それには、

B294H

よりも値を小さくしてみればわかります。

CLEAR ,&HB293 ⏎

とキーインしてみてください。次のようにエラーとなります。しかし、元のようにB294HでならCLEAR文が設定できます。

```
CLEAR ,&HB293
Out of memory
Ok
■
```

【図4・27】──B293Hはエラー

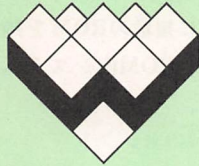
```
CLEAR ,&HB293
Out of memory
Ok
CLEAR ,&HB294

Ok
■
```

B293Hは無理!

B294HならOK!

【図4・28】──B294HはOK

テキスト画面の制御

5

本章より、PC-8801で利用できる基本的な周辺機器の制御を扱います。方法は、主として

ROM内システム・サブルーチン

を利用します。ところで、PC-8801では同一アドレス上に複数のROMを持っています。したがって、ROM内システム・サブルーチンの利用には、

ROMのセレクト

の知識が必要です。本章以外では、特にそのことについて言及していません。適宜、前章以前を参照してください。

以下の構成は、種々の基本的周辺機器がアット・ランダムに出てきます。必要に応じて前後・取捨選択をされて構いません。まずはマシン語による周辺機器の制御、

テキスト画面

から始めます。

5・0 テキスト画面への出力

PC-8801における、マン・マシン・インターフェースは、

CRTディスプレイ

キーボード

となっています。通常、オペレーションはこの2つを通して行なわれます。人間の意志は、キーボードを通してPC-8801に伝えられるし、また、PC-8801からのメッセージはCRTディスプレイを通して伝えられます。

この2つの基本的な周辺機器のうち、最初に

CRTディスプレイの制御

に挑戦することにいたします。CRT画面は、

テキスト画面

グラフィック画面

の2種類があります。我々が最初に挑むのは、もちろん

テキスト画面の制御

です。

1キャラクタの出力

テキスト画面の制御――その最も基本になるのは、

1キャラクタの出力

です。まず、そのサブルーチンから見てみましょう。

<1キャラクタの出力>

アドレス 3E0DH

入　力 Aレジスタ＝キャラクタ・コード

機　能

Aレジスタに出力したい文字のキャラクタ・コードを入れてCALLすると、CRT画面に出力される。コントロール・コードは、その機能が出力される。

このサブルーチンを用いることで、好きな文字をCRTディスプレイに出力することができます。

実例をお目にかけましょう。次の図5・0をご覧ください。

Aレジスタ＝F4H

　　　　　　　└―― "日"のキャラクタ・コード

をセットしています。そして、

CALL　3E0DH

を実行しています。これにより、

日

のキャラクタがCRTディスプレイに表示されます。最後の

RST　　38H

は、〈ブレーク・ポイント〉のところで詳しく分析しましたように

マシン語モニタのホット・スタート

へ戻ります。

```
                    ;=============================
                    ;  PRINT(OUT CRT):1983.7.1
                    ;-----------------------------
                    ;   IN:A=ｷｬﾗｸﾀｰ ｺｰﾄﾞ
                    ;   ﾚｼﾞｽﾀｰ OK
                    ;=============================
                    ;
0038                MON:    EQU  38H
3E0D                PR:     EQU  3E0DH
                    ;
                            ORG  0BB00H
                    ;
BB00 3EF4           EX:     LD   A,'日'
BB02 CD0D3E                 CALL PR
BB05 FF                     RST  MON
                    ;
                            END
```

【図5・0】——1キャラクタの出力

それでは、このプログラムを走らせてみます。次が、そのダンプ・リストです。このマシン・コードを入力してください。

```
h]DBB00,BB05
BB00 3E F4 CD 0D 3E FF
h]■
```

【図5・1】——ダンプ・リスト

そして、

GBB00

と入力し、

```
h]GBB00
```

【図5・2】——Gコマンドの入力

⏎キーを押します。次の図5・3のように

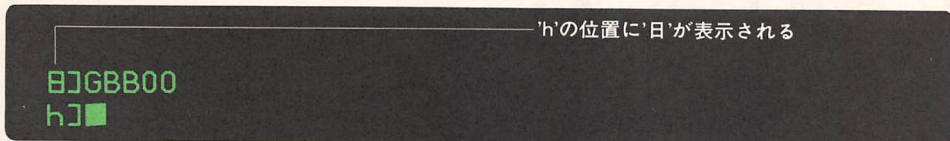
日

のキャラクタが、

hの位置

に表示されます。

'h'の位置に'日'が表示される

```
日コGBB00
hコ■
```

【図5・3】――〝日″の表示

これは、Gコマンド実行直後に、カーソルが〝h″の位置に移るからです。3E0DHのサブルーチンは、

カーソルの位置にキャラクタを表示

する機能を持っているというわけです。

(注)Gコマンドを実行すると、カーソルはその行の先頭に戻りますが、改行は行なわれません。これは、PC-8001のモニタとは異なっています。

コントロール・コードの出力

3E0DH＝1キャラクタの出力

の説明のところを見ますと、

「コントロール・コードは、その機能が出力される」

となっています。これは、次のような意味です。

普通は、Aレジスタに

キャラクタ・コード:20H～FFH

を入れてCALLします。ところが、Aレジスタに

コントロール・コード:00H～1FH

を入れてCALLしますと、CRTディスプレイにキャラクタ・コードが出力される代わりに、その

コントロール・コードの機能

が実行されるというわけです。主なコントロール・コードを図5・4に示します。

それでは、実際にコントロール・コードを使ってみます。図5・3では、〝日″が〝h″の位置に表示されてしまいました。そこで、

コントロール・コード＝0AH

を使ってカーソルを次の行に移し、それからキャラクタ・コードの表示をおこなってみます。そして、今度は

♠ ♥ ◆ ♣

を表示させてみましょう。

プログラムは、図5・5のようになります。

〈コントロール・コード〉	〈機　能〉
01H	HELPキーと同じ
02H	コントロールBと同じ
03H	STOPキーと同じ
05H	コントロールEと同じ
06H	コントロールFと同じ
07H	ブザーを鳴らす
08H	コントロールHと同じ
09H	TABキーと同じ
0AH	Line Feed（改行） カーソルが直下に移動する
0BH	HOMEキーと同じ
0CH	CLRキーと同じ テキスト画面が消去され、カーソルがHOMEの位置に移動する
0DH	Carriage Return（復帰） カーソルが、行の先頭に移動する
0FH	コントロールOと同じ
13H	コントロールSと同じ
15H	コントロールUと同じ
18H	コントロールXと同じ
1BH	ESCキーと同じ

【図5・4】――おもなコントロール・コード

```
                   ;============================
                   ;  PRINT2(OUT CRT):1983.7.1
                   ;----------------------------
                   ;   IN:A=ｷｬﾗｸﾀｰ ｺｰﾄﾞ
                   ;   ﾚｼﾞｽﾀｰ OK
                   ;============================
                   ;
0038               MON:    EQU   38H
3E0D               PR:     EQU   3E0DH
                   ;
                           ORG   0BB00H
                   ;
BB00 3E0A          EX:     LD    A,0AH              ;LINE FEED
BB02 CD0D3E                CALL  PR
BB05 3EE8                  LD    A,'♠'
BB07 CD0D3E                CALL  PR
BB0A 3EE9                  LD    A,'♥'
BB0C CD0D3E                CALL  PR
BB0F 3EEA                  LD    A,'♦'
BB11 CD0D3E                CALL  PR
BB14 3EEB                  LD    A,'♣'
```

```
BB16 CD0D3E          CALL  PR
BB19 FF              RST   MON
          ;
                     END
```

```
h]DBB00,BB19
BB00 3E 0A CD 0D 3E 3E E8 CD 0D 3E 3E E9 CD 0D 3E 3E
BB10 EA CD 0D 3E 3E EB CD 0D 3E FF
h]■
```

【図5・6】――ダンプ・リスト

これが、そのダンプ・リストです。このようにマシン・コードを 入力してください。そして、

GBB00 ⏎

でプログラムの実行開始です。

```
h]GBB00
♠♥♦♣
h]■
```

コントロール・コードによる改行が行なわれている

【5・7】――プログラムの実行

ご覧のように

♠ ♥ ♦ ♣

が表示されました。コントロール・コードが働き、

h]GBB00

の表示が残っていることを確認してください。

5·1 メッセージとコントロール・コード

テキスト画面への出力は、

CALL　3E0DH

でできることはわかりました。ところがメッセージを表示する時のように、たくさんのキャラクタ・コードを表示させるには、特別なサブルーチン等を作って対処する必要があります。しかし、

N88-BASIC　ROM

の中には、メッセージを表示するサブルーチンがあります。我々も、そのサブルーチンを利用させていただくことにいたしましょう。

メッセージ出力ルーチン

メッセージ出力ルーチンは、次のとおりです。

〈メッセージの出力〉

アドレス　5550H

入　力　HLレジスタ＝データ・エリアの先頭アドレス

機　能

出力したいメッセージのデータを

キャラクタ・コードの列＋00H

のように用意し、その先頭アドレスをHLレジスタにセットしてCALLすると、用意したメッセージが表示される。

それでは、実例です。次の図5·8をご覧ください。

これは、メッセージ表示ルーチンを用いて

```
============
  PC-8001
============
```

を表示させようとするものです。サブルーチン5550Hは、

コントロール・コードの出力

も可能ですから、BB07Hで

コントロール・コード＝0AH

を出力して改行しています。また、各行の終りには、

0AH――改行(カーソルを真下へ)

0DH――復帰(カーソルを行の先頭へ)

を出力して行換えを行なっています。

```
                    ;================================
                    ;   5550H
                    ;     MESSAGE:1983.7.1
                    ;--------------------------------
                    ;     IN:HL=DATA AREA(END=00H)
                    ;     ﾚｼﾞｽﾀｰ ?
                    ;================================
                    ;
0038                MON:    EQU  38H                ;MONITOR HOT START
5550                MSG:    EQU  5550H
                    ;
                            ORG  0BB00H
                    ;
BB00 2107BB         EX:     LD   HL,DMSG
BB03 CD5055                 CALL MSG
BB06 FF                     RST  MON
                    ;
BB07 0A             DMSG:   DB   0AH
BB08 3D3D3D3D               DB   '==========',0AH,0DH
BB0C 3D3D3D3D
BB10 3D3D3D0A
BB14 0D
BB15 20205043               DB   '  PC-8001  ',0AH,0DH
BB19 2D383030
BB1D 3120200A
BB21 0D
BB22 3D3D3D3D               DB   '==========',00H
BB26 3D3D3D3D
BB2A 3D3D3D00
                    ;
                            END
```

```
h]DBB00,BB2D
BB00 21 07 BB CD 50 55 FF 0A 3D 3D 3D 3D 3D 3D 3D 3D
BB10 3D 3D 3D 0A 0D 20 20 50 43 2D 38 30 30 31 20 20
BB20 0A 0D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 00
h]■
```

【図5·9】——ダンプ・リスト

それでは実行してみます。このダンプ・リストにしたがってプログラムを入力してください。そして、

GBB00 ⏎

でプログラム・スタートです。

次のように、準備したメッセージが表示されます。改行もきちんと行なわれていることがわかります。コントロール・コードがうまく働いたわけですね。

【図5・10】――プログラムの実行

まとめ

以上のように、テキスト画面への出力は、

3E0DH――1キャラクタの出力
5550H――メッセージの出力

の2つのサブルーチンをうまく利用することで行うことができます。その際、ねらい通りの画面を作るためには、

コントロール・コードの活用

が重要になります。両サブルーチンは、このコントロール・コードを出力できます。いろいろ実験を試み、その使い方を良くマスターしてください。

5·2 画面クリアとカーソル制御

1キャラクタの出力
メッセージの出力

が可能となりました。しかし、これだけでは

テキスト画面

を自由に操作することはできません。たとえば

カーソル位置の制御

ができなければ、画面上の自由な位置に自由なメッセージを表示できないわけです。

これらテキスト画面にまつわる制御の方法を調べて行くことにいたしましょう。

テキスト画面の消去

まずは、簡単なところから

テキスト画面の消去

からまいりましょう。これについては、コントロール・コードを用いて、

```
LD    A,0CH    ←――― 画面消去のコントロール・コード
CALL  3E0DH    ←――― 1キャラクタの出力
```

でも可能です。しかし、N88-BASIC ROMの中には、〈テキスト画面の消去〉のためのサブルーチンが用意されています。せっかくですから、これも利用いたしましょう。

〈テキスト画面の消去〉

アドレス 5F0EH

機能

入力パラメータは、特になし。

このサブルーチンをCALLすることで、テキスト画面を消去できる。

レジスタは、すべて破壊される。

このサブルーチンを用いて、図5·8のプログラムを書き換えたのが、次の図5·11です。最初にテキスト画面を消去した後に、図5·10と同じメッセージを表示しようとしています。

なお、

CALL　5F0EH

により、画面が消去されると同時に、

カーソルがHOMEに移動
LOCATE 0, 0 の位置

しますから、最初に改行する必要はありません。

```
                ;=================================
                ;  5F0EH
                ;    CLS2:1983.7.1
                ;---------------------------------
                ;    ﾚｼﾞｽﾀｰ  :ALL X
                ;=================================
                ;
0038            MON:    EQU  38H                  ;MONITOR HOT START
5550            MSG:    EQU  5550H
5F0E            CLS:    EQU  5F0EH
                ;
                        ORG  0BB00H
                ;
BB00 CD0E5F     EX:     CALL CLS
BB03 210ABB             LD   HL,DMSG
BB06 CD5055             CALL MSG
BB09 FF                 RST  MON
                ;
BB0A 3D3D3D3D DMSG:     DB   '===========',0AH,0DH
BB0E 3D3D3D3D
BB12 3D3D3D0A
BB16 0D
BB17 20205043           DB   '  PC-8001  ',0AH,0DH
BB1B 2D383030
BB1F 3120200A
BB23 0D
BB24 3D3D3D3D           DB   '===========',00H
BB28 3D3D3D3D
BB2C 3D3D3D00
                ;
BB30                    END
```

【図5・11】――テキスト画面の消去

次が、そのダンプ・リストです。これにしたがってマシン・コードを入力してください。

```
h]DBB00,BB2F
BB00 CD 0E 5F 21 0A BB CD 50 55 FF 3D 3D 3D 3D 3D 3D
BB10 3D 3D 3D 3D 3D 0A 0D 20 20 50 43 2D 38 30 30 31
BB20 20 20 0A 0D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 00
h]■
```

【図5・12】――ダンプ・リスト

そして、

GBB00 ⏎

でプログラムをスタートさせます。

```
============
  PC-8001
============
h]■
```

【図5・13】――プログラムの実行

ご覧のように図5・10と同じメッセージが表示されました。しかし、今度は一度画面が消去されてから表示されていますので、

h]GBB00

は、跡形もなく消え去っています。

カーソルの位置制御

次が、いよいよ、

カーソル制御

=カーソルの位置を自由に設定

です。BASICでいえば、

LOCATE X,Y

です。

カーソルを制御するには、次の2つのシステム・ワーク・エリアを書き換えます。

<カーソル位置の制御>

X座標――EF86H(01H～19H)

Y座標――EF87H(01H～50H)

(注)LOCATEと異なり、パラメータは

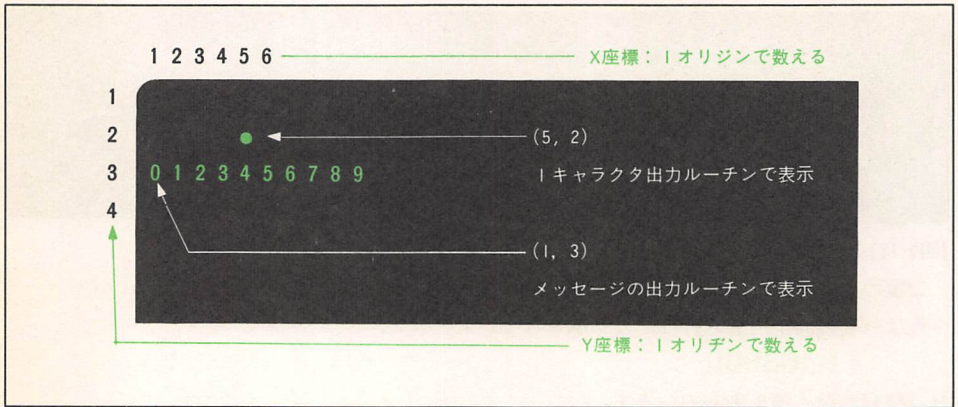
1オリジン(1からカウント)

で与えること!

これをマシン語の命令で表わせば

```
LD      H,<X座標>  }
LD      L,<Y座標>  }――――レジスタ・ペアを使っても可
LD      (EF86H),HL
```

のようにすれば良いでしょう。

次のように表示するプログラムを考えてみます。

【図5・14】——画面の設計

次が、そのプログラムです。

```
                 ;============================
                 ;  CURSOR CONTROL:1983.7.1
                 ;============================
                 ;
0038             MON:   EQU  38H              ;MONITOR HOT START
3E0D             PR:    EQU  3E0DH
5550             MSG:   EQU  5550H
EF86             WCSRY: EQU  0EF86H           ;1-25
EF87             WCSRX: EQU  0EF87H           ;1-80
                 ;
                        ORG  0BB00H
                 ;
BB00 210205      MAIN:  LD   HL,0502H         ;LOCATE 4,1
BB03 2286EF             LD   (WCSRY),HL
BB06 3EEC               LD   A,'●'
BB08 CD0D3E             CALL PR
BB0B 210301             LD   HL,0103H         ;LOCATE 0,2
BB0E 2286EF             LD   (WCSRY),HL
BB11 2118BB             LD   HL,DMSG
BB14 CD5055             CALL MSG
BB17 FF                 RST  MON
                 ;
BB18 30313233 DMSG:     DB   '0123456789',0
BB1C 34353637
BB20 383900
                 ;
                        END
```

【図5・15】——カーソルの位置制御

プログラムは、次のようになっています。

BB00H～BB0AH

1キャラクタ出力ルーチンにより、

(5,2)の位置に"●"

を表示。

BB0BH～BB16H

メッセージ出力ルーチンにより、

(1,3)の位置から"0123456789"

を表示。

```
hJGBB00
BB00 21 02 05 22 86 EF 3E EC CD 0D 3E 21 03 01 22 86
BB10 EF 21 18 BB CD 50 55 FF 30 31 32 33 34 35 36 37
BB20 38 39 00
hJ■
```

【図5・16】――ダンプ・リスト

これが、そのダンプ・リストです。このマシン・コードを入力しましたら、

GBB00 ⏎

でプログラムを走らせます。

【図5・17】――プログラムの実行

ご覧のように、図5・14で設計したとおりの画面が出現しました。

5·3 テキスト画面のコントロール

テキスト画面の自由な位置に、自由に表示できるようになりました。ここまで来れば、もう1つ。

テキスト画面のコントロール

にも挑戦したいですね。すなわち、

テキストのサイズ
スクロール範囲
白黒、カラーのコントロール
ファンクション・キー・ディスプレイの制御

なのです。ここまでマシン語で操作できれば、おおむね不自由はしないでしょう。

テキスト画面のコントロール

テキスト画面のコントロールでしたら、次のサブルーチン1つで間に合います。

〈テキスト画面のコントロール〉

アドレス	6F6BH
入力	Bレジスタ＝画面サイズ横
	Cレジスタ＝画面サイズ縦
機能	

BCレジスタに画面サイズをセットしてCALLすると、テキスト画面の大きさが再設定される。同時にワーク・エリアにセットされた機能が設定される。

このサブルーチンは、テキスト画面のサイズを決定するものですが、同時に、テキスト画面関係のワーク・エリアにセットされた機能も決定することができます。

ワーク・エリアにセットできる機能は、図5・18のとおりです。

たとえば、

E6B9H＝FFH

- FFH：カラーにセット
- E6B9H：白黒・カラーの各モードの切り換え

のようにセットして、

CALL　6F6BH

とすれば、テキスト画面がカラー・モードに切り換わります。

〈アドレス〉	〈機　能〉
E6B2H	スクロール開始行(０１Ｈ～１９Ｈ)
E6B3H	スクロール終了行(０１Ｈ～１９Ｈ)
E6B4H	アトリビュート・コード
E6B5H	ヌル・キャラクター
E6B6H	コントロール・コード・スイッチ
	００Ｈ＝コントロール・コードを表示
	ＦＦＨ＝コントロール・コードを出力
E6B8H	ファンクション・キー表示スイッチ
	００Ｈ＝表示しない
	ＦＦＨ＝表示する
E6B9H	カラー／白黒スイッチ
	００Ｈ＝白黒モード
	ＦＦＨ＝カラー・モード

【図5・18】――テキスト画面コントロール用ワーク・エリア

次に、実際にこのサブルーチンを使ってみます。次のマシン語をキーインしてください。

```
h]DBB00,BB21
BB00 01 14 28 21 B2 E6 36 01 23 36 14 23 36 A8 23 36
BB10 EA 23 3E FF 77 3C 23 23 77 23 36 FF CD 6B 6F C3
BB20 1A 4B
h]■
```

【図5・19】――ダンプ・リスト

そして、

GBB00 ⏎

でプログラムをスタートさせます。次のように妙な記号が表示されて、実行が終了します。

```
Ok CRLF
■
```

【図5・20】――プログラムの実行終了

"CR"とか"LF"

とか、面白い記号が表示されました。何が起こったのでしょう？

```
                ;===================================================
                ;  CRT SET:1983.7.1
                ;---------------------------------------------------
                ;   IN:B=WIDTH-X,C=WIDTH-Y
                ;        (40=28H,80=50H,20=14H,25=19H)
                ;      WSCS=ｽｸﾛｰﾙ ｶｲｼ(1-25)
                ;      WSCE=ｽｸﾛｰﾙ ｼｭｳﾘｮｳ(1-25)
                ;      WATR=ｱﾄﾘﾋﾞｭｰﾄ ｺｰﾄﾞ
                ;      WNUL=ﾇﾙ ｷｬﾗｸﾀｰ ｺｰﾄﾞ
                ;      WCTR=ｺﾝﾄﾛｰﾙ ｺｰﾄﾞ(FFH=ﾋｮｳｼﾞ,00=ﾋｮｳｼﾞ ｼﾅｲ)
                ;      WFKY=ﾌｧﾝｸｼｮﾝ ｷｰ(FFH=ﾋｮｳｼﾞ,00=ﾋｮｳｼﾞ ｼﾅｲ)
                ;      WCOL=ｶﾗｰ ｺｰﾄﾞ(FFH=ｶﾗｰ,00=ｼﾛｸﾛ)
                ;===================================================
                ;
4B1A            BAS:    EQU  4B1AH
6F6B            CRTSET:EQU  6F6BH
E6B2            WSCS:   EQU  0E6B2H
E6B3            WSCE:   EQU  0E6B3H
E6B4            WATR:   EQU  0E6B4H
E6B5            WNUL:   EQU  0E6B5H
E6B6            WCTR:   EQU  0E6B6H
E6B8            WFKY:   EQU  0E6B8H
E6B9            WCOL:   EQU  0E6B9H
                ;
                        ORG  0BB00H
                ;
BB00 011428     EX:     LD   BC,2814H          ;WIDTH 40,20
BB03 21B2E6             LD   HL,WSCS
BB06 3601               LD   (HL),1            ;ｽｸﾛｰﾙ ｶｲｼ
BB08 23                 INC  HL
BB09 3614               LD   (HL),20           ;ｽｸﾛｰﾙ ｼｭｳﾘｮｳ
BB0B 23                 INC  HL
BB0C 36A8               LD   (HL),0A8H         ;ｼｱﾝ
BB0E 23                 INC  HL
BB0F 36EA               LD   (HL),'◆'          ;ﾇﾙ ｷｬﾗｸﾀｰ
BB11 23                 INC  HL
BB12 3EFF               LD   A,0FFH
BB14 77                 LD   (HL),A            ;ｺﾝﾄﾛｰﾙｺｰﾄﾞ ﾋｮｳｼﾞ
BB15 3C                 INC  A                 ;A=00H
BB16 23                 INC  HL
BB17 23                 INC  HL
BB18 77                 LD   (HL),A            ;ﾌｧﾝｸｼｮﾝｷｰ ﾋｮｳｼﾞｼﾅｲ
BB19 23                 INC  HL
BB1A 36FF               LD   (HL),0FFH         ;ｶﾗｰ
BB1C CD6B6F             CALL CRTSET
BB1F C31A4B             JP   BAS
                ;
                        END
```

【図5・21】――逆アセンブル・リスト

それでは、この逆アセンブル・リストにしたがって説明致しましょう。

まず、

Bレジスタ＝２８Ｈ(10進数で40)

Cレジスタ＝１４Ｈ(10進数で20)

にセットしてからCALLしていますから、

画面サイズ＝40×20

にセットされています。文字の大きさが、大きくなったので確認できるでしょう。

また、ファンクション・キーの表示が消えています。これは、

Ｅ６Ｂ８Ｈ＝００Ｈ

にセットしているからです。次に、

Ｅ６Ｂ６Ｈ＝ＦＦＨ

の意味です。ホーム・クリアキーを押してみてください。画面が消える代わりに、

C_L

という記号が現われるでしょう。これは、実は

キャラクタ・コード＝０ＣＨ

のキャラクタです。図5・4をもう一度ご覧ください。０ＣＨは、

キャラクター・コード＝"C_L"

コントロール・コード＝画面をクリアする

の２面性を持っています。Ｅ６Ｂ６Ｈは、その選択機能を持ちます。以下、操作性を良くするためモニタのＳコマンドで

Ｅ６Ｂ６Ｈ＝００Ｈ

に戻してから実験を続けてください。

Ｅ６Ｂ２Ｈ＝０１Ｈ(10進で1)

Ｅ６Ｂ３Ｈ＝１４Ｈ(10進で20)

ですので、スクロール範囲が、１行目～20行目にセットされています。これは、リストなどをとってスクロールさせてみればわかります。

Ｅ６Ｂ９Ｈ＝ＦＦＨ

ですからカラー・モードになっています。画面が水色になっているので、それがわかるでしょう。これは、アトリビュート・コードを

Ｅ６Ｂ４Ｈ＝Ａ８Ｈ(水色)

にセットしたからです。アトリビュート・コード(カラー・モードの場合)については、図5・23をご覧ください。

次に、コントロールＥを押してみてください。

【5・22】――コントロールＥを押すと

押したとたんに、ビックリするでしょう。普通なら、カーソルから右が消去されるはずなのに、ご覧のように◆の大群が現われますから、

〈カラー〉	〈キャラクター・モード〉	〈グラフィック・モード〉
黒	0 8 H	1 8 H
青	2 8 H	3 8 H
赤	4 8 H	5 8 H
紫	6 8 H	7 8 H
緑	8 8 H	9 8 H
シアン	A 8 H	B 8 H
黄	C 8 H	D 8 H
白	E 8 H	F 8 H

【図5・23】――アトリビュート・コード(カラー・モードの場合)

これは、ヌル・キャラクタ・コードとして

E 6 B 5 H＝E A H("◆")

を設定したからです。

BASICへ戻る

最後に、このプログラムではBASICに戻って終了しています。

今まででしたら、

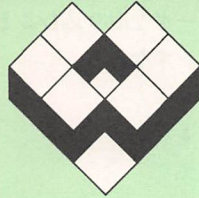
RST　3 8 H

で、マシン語モニタに戻っていたのに。

マシン語のプログラムから、BASICのコマンド・レベルに戻るにはいろいろな方法がありますが、次が最も簡単でしょう。

〈マシン語からBASICへ戻る〉

```
JP　　4 B 1 A H
```

キーボード・スキャニング

マシン語による周辺機器の利用――本章では、

キーボードからのデータ入力

を扱います。これにより、外部からプログラム内部へデータを取り込むことができるようになります。すなわち、

データ入力

↓

内部処理

↓

処理結果の出力

という、プログラムの基本的な流れをこなすことができるようになるわけです。

最初はそれを

静的：入力があるまで待つ

に行います。続いて、

動的：リアルタイム・キー入力

の扱い方も調べます。これにより、キー入力と連動したダイナミックなプログラムの作成が可能になります。

本章の後半では、ある時は災いとなる

キーの先行入力対策

を考えます。そして、最後にはPC-8001でもおなじみの

ファンクション・キーによる

自動オペレーション

に挑戦します。せいぜいイタズラにご利用いただき、マシン語をお楽みください。

6・0 キーボードからの入力

前章では、テキスト画面への出力方法をいろいろ見てきました。これにより、プログラム内部から、CRTディスプレイにメッセージを表示できるようになったわけです。ところが、

外部からのデータ
↓ ―――― 取り込む
プログラム内部

ことは、まだできません。

そこで、最初に〈キーボード〉からの入力データをキャッチする方法を調べて行きます。これにより、

キーボードからデータ入力
内部処理
結果をCRTディスプレイに出力

という基本的処理を行うことが可能になります。

1文字入力ルーチン

キーボードからのデータ入力では、まず次のサブルーチンが基本となります。

〈キーボードより1文字入力〉

アドレス	3583H
出　力	Aレジスタ＝入力データ
機　能	

キーボードから入力があるまで待つ。入力があると、そのキャラクター・コードがAレジスタに入る。

BASICで言えば、INPUT文に相当するものです。ただし、1文字しか扱いませんので、INPUT$(1)と考えた方が良いかもしれません。

実例をお目にかけましょう。

```
h]DBB00,BB11
BB00 3E 0A CD 0D 3E CD 83 35 FE 1B CA 1A 4B CD 0D 3E
BB10 18 F3
h]■
```

【図6・0】――ダンプ・リスト

```
                ;==============================
                ;     INPUT(WAIT) : 1983.7.8
                ;------------------------------
                ;      OUT=A
                ;      ﾚｼﾞｽﾀｰ    Ok
                ;==============================
                ;
3583            INPUT: EQU  3583H
3E0D            PR:    EQU  3E0DH
4B1A            BAS:   EQU  4B1AH                ;BASIC HOT START
                ;
                       ORG  0BB00H
                ;
BB00 3E0A       MAIN:  LD   A,0AH
BB02 CD0D3E            CALL PR                   ;LINE FEED
BB05 CD8335     MA1:   CALL INPUT
BB08 FE1B              CP   1BH                  ;ESC ?
BB0A CA1A4B            JP   Z,BAS
BB0D CD0D3E            CALL PR
BB10 18F3              JR   MA1
                ;
                       END
```

【図6･1】――アセンブル･リスト

これが、そのリストです。

プログラムの意味は、最初にコントロール･コード０ＡＨを出力して改行し、３５８３ＨをCALLして、１文字入力します。それが、

１ＢＨ―― ESC キー

ならば、４Ｂ１ＡＨにジャンプし、BASICに戻ります。それ以外のキーが押されていた時は、３Ｅ０ＤＨをCALLし、そのキャラクタを表示します。そして、またキーの入力を続けます。

こうして、ESC キーが押されるまで

キーボードから１文字入力

↓

そのキャラクタを表示

ということを続けます。

それでは、

GBB00 ⏎

でそれを確めましょう。

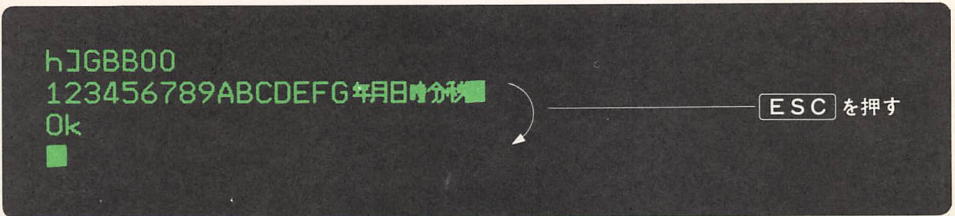
```
h]GBB00
123456789ABCDEFG年月日時分秒■
```

押されたキーが表示される

【図6･2】――プログラムの実行

ご覧のように押されたキーが、次々に表示されていきます。そして、

ESC キー

を押してみてください。

【図6・3】── ESC で終了

ご覧のように、BASICに戻ってプログラムが終了いたします。

スクリーン・エディット・ルーチン

1文字入力ルーチンは、

1文字しか入力できない

CRTに表示されない

などの不便な面があります。一度にたくさんのデータをキーボードから入力したい──そんな時には、次のサブルーチンを利用すると良いでしょう。これにより、

スクリーン・エディット

が可能になります。

〈キーボードより1行入力〉

アドレス 5F92H

出力 Aレジスタ＝最後に入力したキャラクター・コード

HLレジスタ＝E9B9H−1

CY＝ 0：⏎ キーで終了 / 1：STOPキーで終了

機能

スクリーン・エディットを行う。⏎かSTOPキーでエディット・モードを抜け出す。最後にカーソルのあった行が、E9B9H以降のバッファに入る。ENDマークは、00H。最後が、⏎ かSTOPキーかは、AレジスタまたはCYフラグにより判定できる。

それでは、実例です。このサブルーチンを利用してスクリーン・エディットを行ない、入力された1行を、メッセージの出力ルーチンを用いて表示させてみましょう。

プログラムは、次のようになります。

```
                   ;=========================================
                   ;  SCREEN EDIT:1983.7.8
                   ;-----------------------------------------
                   ;    OUT: A=INPUT CODE AT LAST TIME
                   ;         HL=E9B9H-1
                   ;         CY(0=RET,1=STOP)
                   ;      E9B9H ﾖﾘ 255 ﾊﾞｲﾄ ｲﾅｲ (LAST=0)
                   ;      ﾚｼﾞｽﾀｰ: ALL X
                   ;=========================================
                   ;
0038               MON:    EQU  38H                ;MONITOR HOT START
3E0D               PR:     EQU  3E0DH
5550               MSG:    EQU  5550H
5F92               EDIT:   EQU  5F92H
                   ;
                           ORG  0BB00H
                   ;
BB00 3E0A          MAIN:   LD   A,0AH
BB02 CD0D3E                CALL PR                 ;LINE FEED
BB05 CD925F                CALL EDIT
BB08 F5                    PUSH AF
BB09 23                    INC  HL
BB0A CD5055                CALL MSG
BB0D F1                    POP  AF
BB0E 3805                  JR   C,MA1
BB10 211CBB                LD   HL,DRET
BB13 1803                  JR   MA2
BB15 212ABB        MA1:    LD   HL,DSTP
BB18 CD5055        MA2:    CALL MSG
BB1B FF                    RST  MON
                   ;
BB1C 0D0A8383      DRET:   DB   0DH,0AH,'▬ R E T ▬',0
BB20 20522045
BB24 20542083
BB28 8300
BB2A 0D0A8383      DSTP:   DB   0DH,0AH,'▬ S T O P ▬',0
BB2E 20532054
BB32 204F2050
BB36 20838300
                   ;
                           END
```

【図6・4】――プログラム・リスト

ここで注意したいのは、スクリーン・エディットから抜け出た時、

HLレジスタ＝〈入力データ列〉－1

を指しているということです。ですから、単に

INC　　HL

とやるだけで、HLレジスタがメッセージの先頭を指すことになります。しかも、最後の００Hも自動的に書き込まれていますから、

CALL　５５５０H

で入力データを表示できます。

ダンプ・リストは、次のようになります。

```
h]DBB00,BB39
BB00 3E 0A CD 0D 3E CD 92 5F F5 23 CD 50 55 F1 38 05
BB10 21 1C BB 18 03 21 2A BB CD 50 55 FF 0D 0A 83 83
BB20 20 52 20 45 20 54 20 83 83 00 0D 0A 83 83 20 53
BB30 20 54 20 4F 20 50 20 83 83 00
h]■
```

【図6・5】——ダンプ・リスト

GBB00

でプログラム・スタートです。適当にスクリーン・エディットを行なってから、

⏎

を押してみてください。

【図6・6】—— ⏎ で終了

ご覧のように、最後に入力した1行が再度表示されます。そして、⏎ でエディット・モードを抜けたことがわかります。もう一度、プログラムを走らせてください。そして、今度は、

STOPキー

を押してみてください。画面は次のようになります。

【図6・7】—— STOPキーで終了

6·1 リアルタイム・キー入力

前章では、非常に

静的なキー入力ルーチン

を扱いました。次は、

動的なキー入力ルーチン

です。すなわち、

リアルタイム・キー入力ルーチン

を扱います。これにより、リアルタイムな、ダイナミックなGAME、あるいはビジネス・ソフトを作ることができます。

リアルタイム・キー入力ルーチン

まずは、リアルタイム・キー入力ルーチンをご紹介いたします。それは、次のようになっています。

〈リアルタイム・キー入力ルーチン〉

アドレス	35CEH
出力	Aレジスタ＝入力キャラクタ・コード
	Zフラグ 0＝キー入力あり / 1＝キー入力なし

このサブルーチンは、CALLされるとすぐに

現在、キーが押されているかどうかをチェック

します。もし、何も押されていない時は、

Zフラグ＝0

でリターンしてきます。押されていれば、そのキャラクタ・コードがAレジスタに入ってリターンしてくるわけです。

リアルタイム・ゲームの基本

それでは、このサブルーチンを用いて

●

を左右に動かすプログラムを作ってみます。

【図6・8】——キー配置

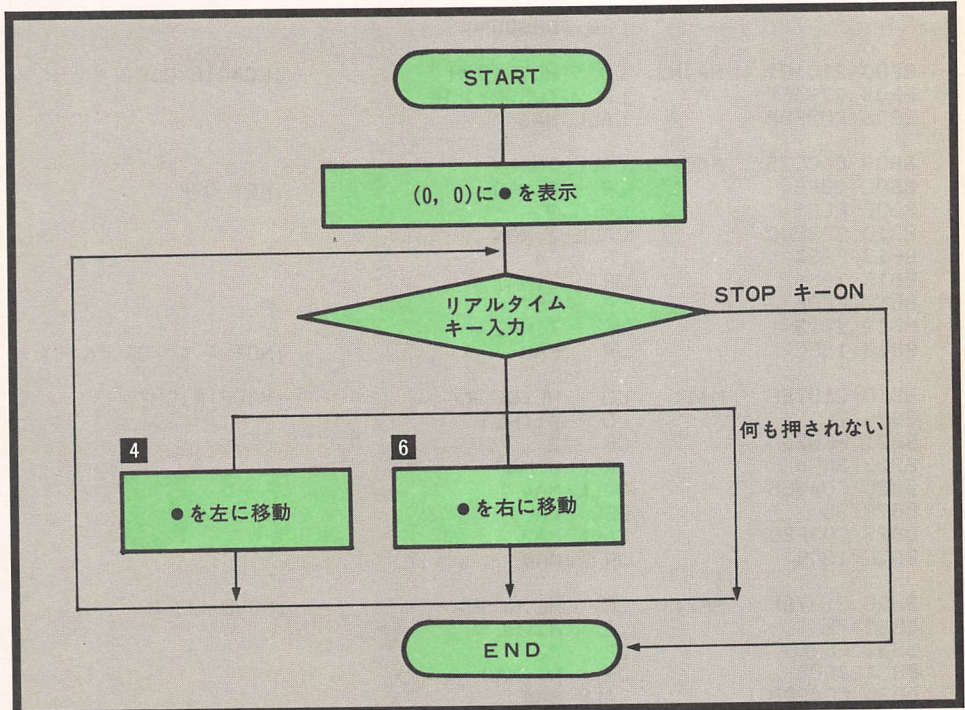
この図のように

4 のキーが押された──→●を左に移動

6 のキーが押された──→●を右に移動

するようにしてみましょう。これは、リアルタイム・ゲームを作る時の最も基本になるところです。

キー入力と図形の移動

画面上の仕様としては、1行目を●が左右に動くように設定してみます。プログラムの流れは、次のようになるでしょう。

【図6・9】──フローチャート

```
h]DBB00,BB4C
BB00 21 01 01 22 86 EF CD 3F BB CD CE 35 28 FB FE 03
BB10 CC 38 00 FE 34 28 06 FE 36 28 13 18 EC 21 87 EF
BB20 7E FE 02 38 E4 CD 43 BB 35 CD 3F BB 18 DB 21 87
BB30 EF 7E FE 28 30 D3 CD 43 BB 34 CD 3F BB 18 CA 3E
BB40 EC 18 02 3E 20 CD 0D 3E 21 87 EF 35 C9
h]■
```

【図6・10】──ダンプ・リスト

```
                ;=================================
                ;  INKEY : 1983.7.8
                ;---------------------------------
                ;   OUT: A=INPUT CODE
                ;        CY(0=KEY ON,1=KEY OFF)
                ;      ﾚｼﾞｽﾀｰ:Ok
                ;=================================
                ;
0038            MON:    EQU  38H                ;MONITOR HOT START
35CE            INKEY:  EQU  35CEH
3E0D            PR:     EQU  3E0DH
EF86            WCSRY:  EQU  0EF86H             ;1-25
EF87            WCSRX:  EQU  0EF87H             ;1-80
                ;
                        ORG  0BB00H
                ;
BB00 210101     MAIN:   LD   HL,0101H           ;LOACTE 0,0
BB03 2286EF             LD   (WCSRY),HL
BB06 CD3FBB             CALL MA3
                ;
BB09 CDCE35     MA0:    CALL INKEY
BB0C 28FB               JR   Z,MA0              ;KEY OFF
BB0E FE03               CP   3
BB10 CC3800             CALL Z,MON
BB13 FE34               CP   ´4´
BB15 2806               JR   Z,MA1
BB17 FE36               CP   ´6´
BB19 2813               JR   Z,MA2
BB1B 18EC               JR   MA0                ;NOT (´4´ OR ´6´ )
                ;
BB1D 2187EF     MA1:    LD   HL,WCSRX           ;MOVE RIGHT
BB20 7E                 LD   A,(HL)
BB21 FE02               CP   2
BB23 38E4               JR   C,MA0
BB25 CD43BB             CALL MA4
BB28 35                 DEC  (HL)
BB29 CD3FBB             CALL MA3
BB2C 18DB               JR   MA0
                ;
BB2E 2187EF     MA2:    LD   HL,WCSRX           ;MOVE LEFT
BB31 7E                 LD   A,(HL)
BB32 FE28               CP   40
BB34 30D3               JR   NC,MA0
BB36 CD43BB             CALL MA4
BB39 34                 INC  (HL)
BB3A CD3FBB             CALL MA3
BB3D 18CA               JR   MA0
                ;
BB3F 3EEC       MA3:    LD   A,´●´              ;PRINT ´●´
BB41 1802               JR   MA5
                ;
BB43 3E20       MA4:    LD   A,´ ´              ;ERASE
BB45 CD0D3E     MA5:    CALL PR
BB48 2187EF             LD   HL,WCSRX
BB4B 35                 DEC  (HL)
BB4C C9                 RET
                ;
                        END
```

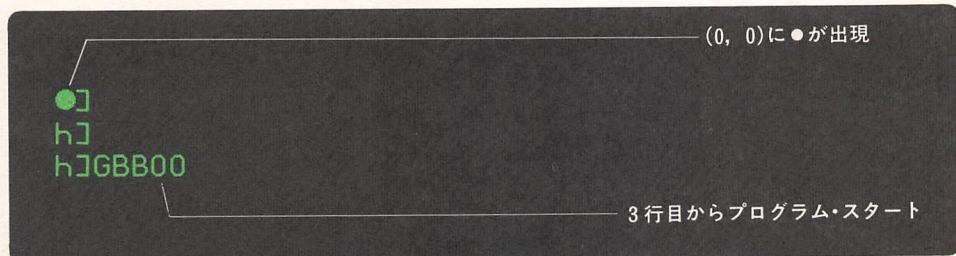
【図6・11】――アセンブル・リスト

図6・10が、ダンプ・リスト。そして、図6・11がアセンブル・リストです。プログラムを入力しましたら、さっそく走らせてみましょう。画面を見やすくするために、

 キー

を押して、一度画面を消去してみてください。そして、⏎ を2回押して、3行目で

GB000 ⏎

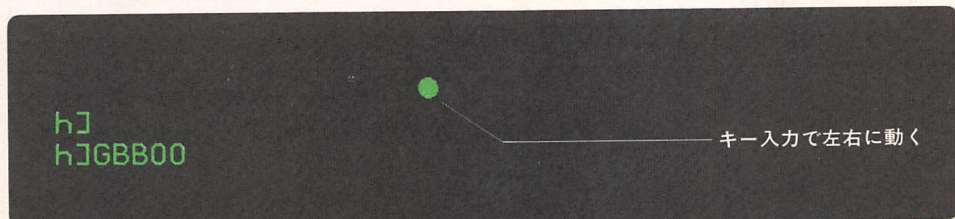
とキーインしてみてください。

【図6・12】――3行目でプログラム・スタート

ご覧のように

●

が(0,0)の位置に表示されます。4 や 6 のキーを押してみてください。次のように

●

が、左右に動くのが確認できるでしょう。

【図6・13】――●が左右に

6・2 キーの先行入力

キー入力については、前節までのサブルーチンでおおむね足りると思います。ところで、PC-8801のキーボードにはあるクセがあります。いわゆる、

キーの先行入力

というやつで、これがなかなかの曲物です。特に、リアルタイムなキー入力を試みようとする時には、案外とこの曲物のために悩まされることがあります。

そこで本節では、この〈キーの先行入力〉という化物を退治することにいたしましょう。

キーの先行入力

さて、まずは

キーの先行入力

です。たぶんご存知だとは思いますが、知らない人のために説明しておきましょう。次のプログラムで体験してみてください。

```
100 PRINT "........................."
110 PRINT "   ";:COLOR 6
120 PRINT "  Please key in !  "
130 COLOR 0
140 PRINT "........................."
150 FOR I=0 TO 10000:NEXT
```

【図6・14】──単純なBASICのプログラム

これは、単純なBASICのプログラムです。

Please key in !

の表示をおこなったのち、

FOR I=0 TO 10000:NEXT

でタイマーを取るだけという単純なものです。

```
RUN
.........................
   Please key in !
.........................
```

リバース
ブリンク
タイマーがきいている

【図6・15】──RUN

ご覧のようにRUNしますと、真中のメッセージが、ブリンク・リバースします。そして、タイマーがきいていますから、しばらくは

Ok

の表示が出ません。この間に

ABCDEFGHIJKLMNOPQRSTU

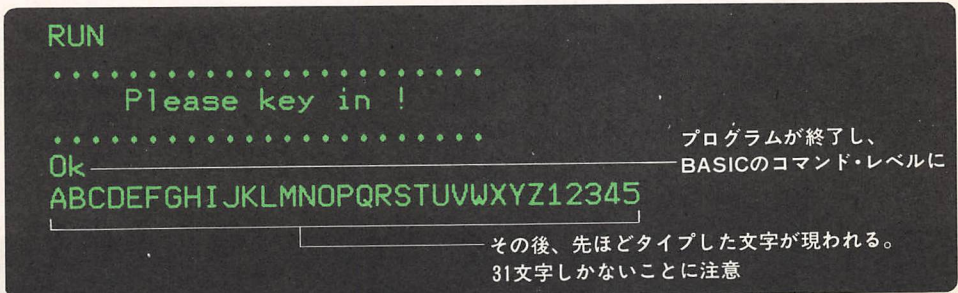
VWXYZ1234567890

の36文字をキーインしてください(こんな短時間では、これだけ打てない？　ダメですねえ。もっとキーボードを愛用いたしましょう)。

やがて、タイマーがきれますと、次のように表示されます。

```
RUN
..........................
    Please key in !
..........................
Ok
ABCDEFGHIJKLMNOPQRSTUVWXYZ12345
```

プログラムが終了し、BASICのコマンド・レベルに

その後、先ほどタイプした文字が現われる。31文字しかないことに注意

【図6・16】――キー先行入力の効果

「Ok」が表示された後、先ほどキーインした文字が現われました。すなわち、〈キーの先行入力〉とは、

プログラム実行中でもキーの入力を受け付ける

というものです。ただし、バッファの関係で受け付けてもらえるのは、

31文字

までです。

先行入力の禁止

〈キー先行入力〉は、速いタイプにもキー入力が追いつくなど便利なものです。しかし、リアルタイム・キー入力の時などには、かえってわずらわしいものです。BASICの命令で、自由に

〈キーの先行入力〉を禁止

できるようになっていれば良かったのですが――。

〈キーの先行入力〉は、システム・ワークエリアの一部を書き換えることで可能になります。

〈キー先行入力禁止スイッチ〉

E6CDH ｛ 00H――――キーの先行入力可能
　　　　　00H以外――――キーの先行入力不可能

このことを利用して、<キーの先行入力>を禁止してみましょう。次のプログラムをご覧ください。

```
10 KYFLAG=&HE6CD:KON=0:KOFF=&HFF ———— 変数定義
100 PRINT "........................"
110 PRINT "   ";:COLOR 6
120 PRINT "   Please key in !   "
130 COLOR 0
140 PRINT "........................"
145 POKE KYFLAG,KOFF ———— 先行入力の禁止
150 FOR I=0 TO 10000:NEXT
160 POKE KYFLAG,KON ———— 先行入力可能に
```

【図6・17】——キー先行入力の禁止

これは、図6・14のプログラムに3行程追加したものです。

RUN ⏎

で走らせてみます。

```
RUN
........................
    Please key in !
........................
```

【図6・18】——プログラム実行中にキーイン

図6・15と同じ画面が現われました。タイマーがきいている間に先ほどと同じように

ABCDEFGHIJKLMNOPQRSTU
VWXYZ1234567890

の36文字をキーインしてください。また、STOPキーを押しても効かないことを確認しておいてください。次のように終了します。

【図6・19】——キー先行入力は効かず

6・3 自動オペレーションの設定

キーボードの制御——最後は、

自動オペレーション

に挑戦してみましょう。

PC-8001では、ファンクション・キーを利用した〈自動オペレーション〉のテクニックが良く使われていました。カセット・レコーダーをオート・スタートさせたり、DATA文を自己増殖させたりといったように——。

PC-8801でも似たようなことができます。しかし、前節で見ました〈キーの先行入力〉を利用しているため、

●31文字までしか定義できない

●一部のコマンドに〈キー先行入力〉をクリアしてしまうものがある

の制限があります。しかし、これでも使い方によっては面白いことができますので、挑戦してみるのも面白いでしょう。

自動オペレーション

実例をお目にかけます。

次のプログラムをキーインしてください。

```
h]DBB00,BB53
BB00 21 45 BB CD 50 55 CD C8 5F 23 11 80 F2 7E 12 13
BB10 23 A7 20 F9 12 1B 3E 0D 12 21 1E F2 22 B7 EE 01
BB20 28 00 11 1E F2 21 2D BB ED B0 C3 1A 4B 3E FF 32
BB30 CD E6 32 CE E6 21 80 F2 22 03 F0 AF 32 0C F0 32
BB40 CD E6 C3 1A 4B 0A 41 55 54 4F 20 43 4F 4D 4D 41
BB50 4E 44 3A 00
h]■
```

【図6・20】——ダンプ・リスト

そうしましたら、

GBB00 ⏎

でプログラム・スタートです。

```
h]GBB00
AUTO COMMAND:■
```

入力待ち

【図6・20】——プログラム・スタート

図のように

AUTO COMMAND:

と表示され、入力待ちとなります。ここで実行したい命令をキーインします。たとえば

LOAD "TEST. KT" ⏎

とキーインしてみましょう。

```
hJGBBOO
AUTO COMMAND:LOAD "TEST.KT"
Ok
■
```

【図6・21】——命令をキーイン

ご覧のように、BASICのコマンド・レベルに戻ってプログラムが終了します。ここで、ドライブ1のディスケットに

"TEST. KT"

のファイルが入っていると仮定して（なければ、適当なファイルをセーブしておいてください）。

CMD ⏎

とキーインしてください。

```
CMD
Ok          ←—— 1度BASICに戻った後
LOAD "TEST.KT"  ←—— 命令が実行される
Ok
■
```

【図6・22】——CM D実行

ご覧のように一度、BASICのコマンド・レベルに戻ります。しかる後に、先ほど定義しておいたコマンドが実行されます。もちろん

"TEST. KT"

が、ロードされます。

プログラム中のステートメントとして

もう1つ例をお目にかけましょう。この自動オペレーションが、

BASICのプログラム中でも使用可能

なことをお見せするためです。

もう一度、

GBB00 ⏎

で図6・20のプログラムを走らせます。そして、今度は、

CONSOLE ,,1 ⏎

と定義してください。

```
MON

hJGBB00
AUTO COMMAND:CONSOLE ,,1 ―――――― 定 義
Ok
■
```

【図6・23】――CONSOLE ,,1を定義

そうしましたら、次のBASICのプログラムをキーインしてください。

```
100 '--------------------
110 '   TEST :1983.7.10
120 '--------------------
130 '
140 CONSOLE ,,0
150 '
160 PRINT "━━━━━━━━━━━━━━━"
170 PRINT "   ナ ツ メ シ ャ   "
180 PRINT "━━━━━━━━━━━━━━━"
190 '
200 CMD ―――――――――――――――― CMD
```

【図6・24】――BASICプログラム・リスト

200行目にCMDの命令が入っていることに注意してください。

RUN ⏎

でBASICのプログラムをスタートさせます。

【図6・25】――RUN実行

まず、BASICのプログラムにより、〈某社〉のメッセージが表示されます。140行目の

CONSOLE

により、ファンクション・キーの表示が消えます。そして、「Ok」の表示が出た後、自動オペレーションが起動して

CONSOLE,,1

が実行され、ファンクションキーが表示されます。

ファンクション・キーのしくみ

それでは、以上の説明です。次ページに、図2・20のアセンブル・リストを示します。

このプログラムは、

BB00H〜BB18H
　オート・コマンドをF280Hからのバッファにストアする。

BB19H〜BB1EH
　CMDのジャンプ・テーブルを書き換える。

BB1FH〜BB29H
　CMD処理ルーチンを、システム・ワークエリアの空きエリアに転送する。

の3つの部分から成っています。

ここで用いたデータは、次のとおりです。

〈CMDジャンプ・テーブル〉

EEB6H――C3H
EEB7H――1EH　JP　F21EH　に書き換え
EEB8H――F2H

【図6・26】――CMDジャンプ・テーブルの書き換え

〈ファンクション・キー・ポインタ〉

F003H
　現在出力中のデータの入っているアドレスを指す。

〈ファンクション・キー・フラグ〉

E6CEH
　00H――――ファンクション・キーON
　00H以外――ファンクション・キーOFF

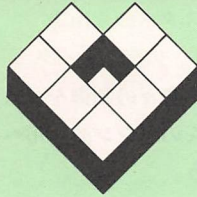
【図6・27】――ファンクション・キーのセッティング

ファンクション・キー・ポインタにデータの先頭アドレスを入れ、ファンクション・キー・フラグを立ててやれば実行されます。

```
                  ;==================================
                  ;  AUTO OPERATION :1983.7.8
                  ;==================================
                  ;
4B1A              BAS:    EQU  4B1AH                    ;BASIC HOT START
5550              MSG:    EQU  5550H
5FC8              INPUT:  EQU  5FC8H
E6CD              WKYOFF:EQU  0E6CDH
E6CE              WKYFLG:EQU  0E6CEH
EEB6              CMD:    EQU  0EEB6H
F003              WKYADR:EQU  0F003H
F00C              WKYNMB:EQU  0F00CH
F21E              WEMPTY:EQU  0F21EH
F280              WCMD:   EQU  0F280H
                  ;
                          ORG  0BB00H
                  ;
BB00 2145BB       MAIN:   LD   HL,D1
BB03 CD5055               CALL MSG
BB06 CDC85F               CALL INPUT
BB09 23                   INC  HL
BB0A 1180F2               LD   DE,WCMD
BB0D 7E           MA1:    LD   A,(HL)                   ;COMMAND SET
BB0E 12                   LD   (DE),A
BB0F 13                   INC  DE
BB10 23                   INC  HL
BB11 A7                   AND  A
BB12 20F9                 JR   NZ,MA1
BB14 12                   LD   (DE),A                   ;SET RETURN KEY
BB15 1B                   DEC  DE
BB16 3E0D                 LD   A,0DH
BB18 12                   LD   (DE),A
BB19 211EF2               LD   HL,WEMPTY                ;SET JUMP TABLE
BB1C 22B7EE               LD   (CMD+1),HL
BB1F 012800               LD   BC,40
BB22 111EF2               LD   DE,WEMPTY
BB25 212DBB               LD   HL,D0
BB28 EDB0                 LDIR
BB2A C31A4B               JP   BAS
                  ;
BB2D 3EFF         D0:     LD   A,0FFH                   ;CMD OPERATION
BB2F 32CDE6               LD   (WKYOFF),A               ;KEY OFF
BB32 32CEE6               LD   (WKYFLG),A               ;F-KEY FLAG ON
BB35 2180F2               LD   HL,WCMD
BB38 2203F0               LD   (WKYADR),HL
BB3B AF                   XOR  A
BB3C 320CF0               LD   (WKYNMB),A
BB3F 32CDE6               LD   (WKYOFF),A
BB42 C31A4B               JP   BAS
                  ;
BB45 0A415554     D1:     DB   0AH,'AUTO COMMAND:',0
BB49 4F20434F
BB4D 4D4D414E
BB51 443A00
                  ;
                          END
```

【図6・28】――アセンブル・リスト

第7章

その他の周辺機器の制御

いよいよ最後の章となりました。残されたテーマは、

カセット・レコーダーの制御

プリンタの制御

ディスクへの入出力

カラー・グラフィック

と多彩です。各々の制御をマスターすることで、たくさんのマシン語応用への道が開けてくるでしょう。

たとえば、〈カセット・レコーダーの制御〉では、

音量調整プログラム

をご紹介いたします。これによりあなたのカセット・レコーダーとPC-8801との相性診断ができます。またこのテクニックを利用することで、

- ●未知の録音テープのデータ・フォーマットを調べるプログラム
- ●未知の録音プログラムをコピーするプログラム

などが作れます。ぜひ、挑戦してみてください。

7·0 カセット・レコーダーの制御

カセット・レコーダーを制御するには、通信に関する知識が必要となります。そして、長いプログラムを書く必要があります。たとえばカセット・レコーダーにデータを書き込もうとすると、次の手順が必要になります。

出力ポートの初期設定

- キャリアの設定
- モーターOFF
- μPD8251CのSIO→CMTの切り換え
- ボーレートの設定(上と同時)

出力ポートをデータ書き込みに

モーターON

キャリア書き込み

～

データの読み込みは、もっと大変です。したがって、これらのルーチンを作るには大変な手間なわけです。

ところで、これらのルーチンは、もちろんROM内に用意されています。そこで我々も、このシステム・サブルーチンを利用することにいたしましょう。以下にその手順を説明いたします。そして実例として、カセット・レコーダーの音量調整に役立つプログラムをご紹介いたします。

カセットのデータ出力

まずは、カセット書き込み用のサブルーチンです。

〈書き込み用初期設定〉

アドレス　７Ｆ４ＤＨ

入　力　Ｆ００９Ｈ　ＦＡＨ――600ボー / ＦＢＨ――1200ボー

〈データの出力〉

アドレス　７ＦＤ０Ｈ

入　力　Aレジスタ＝出力データ

〈書き込み後処理ルーチン〉

アドレス　７Ｆ１ＡＨ

【図7·0】――カセット書き込み用サブルーチン

次のプログラムをご覧ください。

```
                  ;==============================
                  ;  WRITE CASSETTE : 1983.7.15
                  ;==============================
                  ;
                          ORG  0BB00H
                  ;
0038              MON:    EQU  38H
7F1A              WCLOSE:EQU  7F1AH
7F4D              WINIT:  EQU  7F4DH                    ;WRITE INITIAL
7FD0              COUT:   EQU  7FD0H                    ;WRITE CASSETTE(A=CODE)
F009              BAUD:   EQU  0F009H                   ;FA=600 baud,FB=1200 baud
                  ;
BB00 3EFA         MAIN:   LD   A,0FAH                   ;600 ﾎﾞｰ
BB02 3209F0               LD   (BAUD),A
BB05 CD4D7F               CALL WINIT
BB08 2135BB               LD   HL,D1
BB0B CD2CBB               CALL SUB
BB0E 3E41                 LD   A,'A'
BB10 061A                 LD   B,26
BB12 0E50         MA1:    LD   C,80
BB14 CDD07F       MA2:    CALL COUT
BB17 0D                   DEC  C
BB18 20FA                 JR   NZ,MA2
BB1A 3C                   INC  A
BB1B 10F5                 DJNZ MA1
BB1D 2185BB               LD   HL,D2
BB20 CD2CBB               CALL SUB
BB23 3E41                 LD   A,'A'                    ;FINISH
BB25 CDD07F               CALL COUT
BB28 CD1A7F               CALL WCLOSE
BB2B FF                   RST  MON
                  ;
BB2C 7E           SUB:    LD   A,(HL)                   ;ｶｷｺﾐ
BB2D A7                   AND  A
BB2E C8                   RET  Z
BB2F CDD07F               CALL 7FD0H
BB32 23                   INC  HL
BB33 18F7                 JR   SUB
                  ;
BB35 0D0A         D1:     DB   0DH,0AH
BB37 2D2D2D2D             DB   '-------------------------',0DH,0AH
BB3B 2D2D2D2D
BB3F 2D2D2D2D
BB43 2D2D2D2D
BB47 2D2D2D2D
BB4B 2D2D2D0D
BB4F 0A
BB50 20204341             DB   '  CASSETTE TEST START  ',0DH,0AH
BB54 53534554
BB58 54452054
BB5C 45535420
BB60 53544152
BB64 5420200D
BB68 0A
BB69 2D2D2D2D             DB   '-------------------------',0DH,0AH,0DH,0AH,0
BB6D 2D2D2D2D
BB71 2D2D2D2D
BB75 2D2D2D2D
BB79 2D2D2D2D
BB7D 2D2D2D0D

BB81 0A0D0A00
BB85 0D0A         D2:     DB   0DH,0AH
BB87 2D2D2D2D             DB   '-----------------------',0DH,0AH
```

```
BB8B 2D2D2D2D
BB8F 2D2D2D2D
BB93 2D2D2D2D
BB97 2D2D2D2D
BB9B 2D0D0A
BB9E 20204341          DB    '   CASSETTE TEST END   ',0DH,0AH
BBA2 53534554
BBA6 54452054
BBAA 45535420
BBAE 454E4420
BBB2 200D0A
BBB5 2D2D2D2D          DB    '---------------------',0DH,0AH,0
BBB9 2D2D2D2D
BBBD 2D2D2D2D
BBC1 2D2D2D2D
BBC5 2D2D2D2D
BBC9 2D0D0A00
                 ;
                       END
```

【図7・1】——書き込み用プログラム

これは、〈カセット音量調整書き込み用プログラム〉です。次がそのダンプ・リストです。

```
h]DBB00,BBCC
BB00 3E FA 32 09 F0 CD 4D 7F 21 35 BB CD 2C BB 3E 41
BB10 06 1A 0E 50 CD D0 7F 0D 20 FA 3C 10 F5 21 85 BB
BB20 CD 2C BB 3E 41 CD D0 7F CD 1A 7F FF 7E A7 C8 CD
BB30 D0 7F 23 18 F7 0D 0A 2D 2D 2D 2D 2D 2D 2D 2D 2D
BB40 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 0D 0A
BB50 20 20 43 41 53 53 45 54 54 45 20 54 45 53 54 20
BB60 53 54 41 52 54 20 20 0D 0A 2D 2D 2D 2D 2D 2D 2D
BB70 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
BB80 0D 0A 0D 0A 00 0D 0A 2D 2D 2D 2D 2D 2D 2D 2D 2D
BB90 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 0D 0A 20 20
BBA0 43 41 53 53 45 54 54 45 20 54 45 53 54 20 45 4E
BBB0 44 20 20 0D 0A 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
BBC0 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 0D 0A 00
h]■
```

【図7・2】——ダンプ・リスト

このマシン・コードを入力し、カセットを録音状態にしてから

GBB00 ⏎

で走らせてください。

カーソル点滅

【図7・3】——データの書き込み

ご覧のようにデータの書き込みが始まります。

やがて、次のように録音が終了します。

```
h]GBB00
h]■
```

【図7·4】——録音終了

カセット入力データをCRTへ

以上の実験で、〈何か〉がカセットに録音されたらしいことはわかります。何が、録音されたのでしょう？

次に、そのデータを読み込むプログラムをご紹介いたします。

```
                    ;==============================
                    ;  READ CASSETTE : 1983.7.15
                    ;==============================
                    ;
                            ORG   0BB00H
                    ;
0038                MON:    EQU   38H
3E0D                PR:     EQU   3E0DH
7ED0                RINIT:  EQU   7ED0H
7F15                RCLOSE:EQU    7F15H
7F87                CIN:    EQU   7F87H
F009                BAUD:   EQU   0F009H
                    ;
BB00 3EFA           MAIN:   LD    A,0FAH                ;600 ﾎﾞｰ
BB02 3209F0                 LD    (BAUD),A
BB05 CDD07E                 CALL  RINIT
BB08 CD877F         MA1:    CALL  CIN
BB0B FE24                   CP    '$'
BB0D 2805                   JR    Z,MA2
BB0F CD0D3E                 CALL  PR
BB12 18F4                   JR    MA1
BB14 CD157F         MA2:    CALL  RCLOSE
BB17 FF                     RST   MON
                    ;
                            END
```

【図7·5】——読み込み用プログラム

このダンプ·リストは、次のとおりです。 これにしたがってデータを入力してください。

```
h]BB00,BB17
BB00 3E FA 32 09 F0 CD D0 7E CD 87 7F FE 24 28 05 CD
BB10 0D 3E 18 F4 CD 15 7F FF
h]■
```

【図7·6】——ダンプ·リスト

このプログラムで用いた、カセット・レコーダー読み込み用のシステム・サブルーチンは、次のとおりです。

〈読み込み用初期設定〉

アドレス	7ED0H	
入　力	F009H	FAH：600ボー FBH：1200ボー

〈データの入力〉

アドレス	7F87H
出　力	Aレジスタ＝入力データ

〈読み込み後処理ルーチン〉

アドレス	7F15H

【図7・7】——カセット読み込み用サブルーチン

マシン語の入力が終わりましたら、カセット・テープを巻き戻し、キャリアの部分まで進めてください。そして、一度カセット・レコーダーを止めます。そして、

GBB00 ⏎

でプログラムをスタートさせます。最後に、もう一度カセット・レコーダーを再生させてください。次のように、カセットから読み込んだデータが、CRTディスプレイ上に表示されます。

```
hJGBB00
------------------------
  CASSETTE TEST START
------------------------

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
FFFFFFFFFFFFFFF
```

【図7・8】——カセット読み込み中

カセットの最適音量

ご覧のように、図7·1のプログラムは、

CASSET TEST START

のメッセージと、

AAAAA……………………AAA
BBBBB……………………BBB
～
ZZZZZ……………………ZZZ

（ヨコ80文字）

のデータをカセットに書き込むものでした。そして、図7·5のプログラムがそのデータを、読み込むためのものだったのです。

```
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK
LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO
PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ

----------------------
  CASSETTE TEST END
----------------------
```

【図7·9】——カセット読み込み終了

これが、無事読み込みが終了した時の画面です。そこで、カセットの読み込み中に、カセットのボリュームをいろいろに変えてみてください。最初は、ボリュームを除々に下げて行きます。適正音量の**下限**を越えたところで誤ったデータを読むか、

Tape read error

が起こるでしょう。その時の目盛を記録しておきます。

同様にして、ボリュームを除々に上げて行った時の**上限**を求めます。この2つの目盛から、

適正音量：〈下限〉～〈上限〉

であることがわかります。

カセット制御の手順

以下に、マシン語でカセットの制御をする際の手順をまとめておきます。必要なサブルーチンは、

出力関係：図7・0

入力関係：図7・7

を参照してください。

【図7・10】――カセット制御の手順

7・1 プリンタの制御

プリンタを制御するには、たくさんの方法があります。ここでは、

プリンタへの1文字出力ルーチン

を使ってみます。これなら、各種コントロール・コード も出力できますから、ほとんどのアプリケーションで使用可能でしょう。

プリンタへの出力ルーチン

ここで用いるシステム・サブルーチンは次のとおりです。

<プリンタへの1文字出力>

アドレス 3ED4H

入　力 Aレジスタ＝キャラクタ・コード

- レジスタはすべて保証
- コントロール・コードを各種プリンタに合わせて出力できる

これを用いたプログラムを次に示します。

```
                 ;==============================
                 ;  PRINTER TEST2: 1983.7.15
                 ;==============================
                 ;
                         ORG   0BB00H
                 ;
0038             MON:    EQU   38H                    ;MONITOR HOT START
3ED4             LP:     EQU   3ED4H
                 ;
BB00 3E41        MAIN:   LD    A,'A'
BB02 0607                LD    B,7
BB04 0E1E        MA1:    LD    C,30
BB06 CDD43E      MA2:    CALL  LP
BB09 0D                  DEC   C
BB0A 20FA                JR    NZ,MA2
BB0C F5                  PUSH  AF
BB0D 3E0D                LD    A,0DH
BB0F CDD43E              CALL  LP
BB12 3E0A                LD    A,0AH
BB14 CDD43E              CALL  LP
BB17 F1                  POP   AF
BB18 3C                  INC   A
BB19 10E9                DJNZ  MA1
BB1B FF                  RST   MON
                 ;
                         END
```

【図7・11】――プリンタ・テスト用プログラム

A ～ G

の7つの英字を、各30キャラクタずつ表示するものです。

PC-8023-Cで改行する時には、

コントロール・コード：0DH

0AH

を必ずこの順序で出力してください。

このプログラムのダンプ・リストを次に示します。これにしたがってマシン・コードを入力してください。

```
hJDBB00,BB1B
BB00 3E 41 06 07 0E 1E CD D4 3E 0D 20 FA F5 3E 0D CD
BB10 D4 3E 3E 0A CD D4 3E F1 3C 10 E9 FF
hJ■
```

【図7・12】——ダンプ・リスト

入力が終りましたら、プリンタのスイッチをONにします。

そして、

GBB00 ⏎

でプログラム・スタートです。次のようなリストが得られるでしょう。

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
GGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
```

【図7・13】——プリンタへの出力

7・2　ディスクへの入出力

マシン語による周辺機器の制御——次は、

ディスクへの入出力

です。ディスクとのやり取りをシステム・サブルーチンを利用して行うには、データを256バイト単位で扱うのが便利です。それには、次のサブルーチンを利用します。

1セクター入出力ルーチンの利用

<1セクター入出力カルーチン>

アドレス　369AH

入　力　Bレジスタ＝トラック番号、面
Cレジスタ＝セクタ番号
HLレジスタ＝入出力されるデータ領域のメモリ・トップ
EC85H＝ドライブ番号－1
CY　1＝書き込み
　　0　読み込み(Z＝0)
　　　　ベリファイ(Z＝1)

出　力　CY　0＝正常終了
　　　　　　1＝エラー

このうち、両面倍密版における

トラック番号
面
　→Bレジスタ

については、説明が必要でしょう。これらについては、次のように指定してください。

サーフィス0(おもて面)
　Bレジスタ＝<トラック番号>×2
サーフィス1(うら面)
　Bレジスタ＝<トラック番号>×2＋1

ディスクからの入力

次に、このサブルーチンを用いて、

ドライブ2
サーフィス1

トラック3

セクター4

のデータをディスケット上から

D000H〜D0FFH

に読み込んでみます。

プログラムは次のようになるでしょう。

```
                 ;========================
                 ;  DISK I/O : 1983.7.15
                 ;========================
                 ;
0038             MON:    EQU  38H                  ;MONITOR HOT START
369A             DISK:   EQU  369AH
EC85             DRV:    EQU  0EC85H               ;DRIVE# (0-3)
                 ;
                         ORG  0BB00H
                 ;
BB00 3E01        MAIN:   LD   A,1                  ;DRIVE=2
BB02 3285EC              LD   (DRV),A
BB05 0607                LD   B,7                  ;SURFACE=1,TRACK=3
BB07 0E04                LD   C,4                  ;SECTOR=4
BB09 2100D0              LD   HL,0D000H            ;MEMORY TOP
BB0C AF                  XOR  A                    ;CY=0
BB0D 3D                  DEC  A                    ;Z=0
BB0E CD9A36              CALL DISK
BB11 FF                  RST  MON
                 ;
                         END
```

【図7・15】――アセンブル・リスト

ダンプ・リストは次のようになります。

```
h]DBB00,BB11
BB00 3E 01 32 85 EC 06 07 0E 04 21 00 D0 AF 3D CD 9A
BB10 36 FF
h]■
```

【図7・16】――ダンプ・リスト

このマシン・コードを入力し、ドライブ2にディスケットをセット
してから、

でプログラムを走らせます。カチン、とディスクをアクセスする音が
聞こえ、プログラムの実行が終了します。

```
h]GBB00    ――――――――――――― ディスクをアクセス
h]■
```

【図7・16】――実行終了

DD000、D0FF ⏎

で、読み込まれたデータを見てみます。

```
h]DD000,D0FF
D000 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22
D010 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
D020 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33
D030 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
D040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D0A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D0B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D0C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D0D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D0E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D0F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
h]■
```

【図7・17】――Dコマンドでデータを確認

これは、あらかじめディスケットにこのように書き込んでおいたからです。実際、図7・15のプログラムが正しいか確認したいのであれば、モニタでコントロールDを押した後、

2, 1, 3, 4 ⏎

とキーインしてください。先ほどと同じ出力結果が得られればOK、というわけです。

7・3　カラー・グラフィック

いよいよ、最後の節となりました。残されたテーマは、

カラー・グラフィックの制御

です。カラー・グラフィックを扱うための有用なルーチンをご紹介いたします。

PC-8801におけるグラフィック関係のサポート・ルーチンは

サブROM

に入っています。サブROMの利用のしかたについては、第4章で詳しく説明いたしました。したがって以下で紹介しますシステム・サブルーチンにつきましては、4・2節を参照してください。

グラフィック関係システム・サブルーチン

もう1度、図4・12をご覧になってください。ここに、サブROMをCALLするための31組のデータが用意されています。このデータの具体的な利用方法につきましては、図4・20のプログラムが役に立つでしょう。

次の図7・18に、この中から有用なものをピック・アップして示します。実際にマシンの上で確認し、使い方をよくマスターしてください。なお、この表における〈アドレス〉は、メインROMからサブROMをCALLする際のアドレスです。直接ユーザー・プログラムからCALLするのに使ってかまいません。

マシン語によるPSET

次に例を示します。

図7・19をご覧ください。PSET文により斜めの直線を描いてみようというものです。初期位置は、

(50, 50)

で、カラーは黄色ですから

```
DB  ´(50,50),6´,0
              └──────── お忘れなく！
```

のデータを用意します。HLレジスタにこのトップ・アドレスをセットしてCALLすればOKです。プログラムでは、

DJNZループ

を作って9回CALLしています。ループごとに点の位置を1ドットずつずらして斜線を作っています。

アドレス	サブルーチン番号	機　能
6E96H	00H	PRESET
66E9AH	01H	PSET
6EA2H	03H	GETⓐ、PUTⓐ 使い分けは、システム・ワークエリア Ｆ０７１Ｈで行う。 Ｆ０７１Ｈ｛００Ｈ──GETⓐ／０１Ｈ──PUTⓐ
6EAEH	06H	LINE
6EBAH	09H	VIEW
6ECEH	0EH	CIRCLE
6ED6H	10H	WINDOW
6EDAH	11H	PAINT
6EDEH	12H	SCREEN
6EE6H	14H	CLS2 グラフィック画面の消去

【図7・18】──サブROM内システム・サブルーチン

```
                 ;=====================
                 ;  PSET : 1983.7.15
                 ;=====================
                 ;
                          ORG   0BB00H
                 ;
0038             MON:     EQU   38H
5F0E             CLS:     EQU   5F0EH
6E9A             PSET:    EQU   6E9AH
                 ;
BB00 CD0E5F      MAIN:    CALL  CLS
BB03 0609                 LD    B,9
BB05 2118BB      MA1:     LD    HL,DATA
BB08 C5                   PUSH  BC
BB09 CD9A6E               CALL  PSET
BB0C C1                   POP   BC
BB0D 211ABB               LD    HL,DATA+2
BB10 34                   INC   (HL)
BB11 211DBB               LD    HL,DATA+5
BB14 34                   INC   (HL)
BB15 10EE                 DJNZ  MA1
BB17 FF                   RST   MON
                 ;
BB18 2835302C    DATA:    DB    '(50,50),6',0
BB1C 3530292C
BB20 3600
                 ;
BB22                      END
```

【図7・19】──PSET文の利用

ダンプ・リストは、次のとおりです。

```
hDBB00,BB21
BB00 CD 0E 5F 06 09 21 18 BB C5 CD 9A 6E C1 21 1A BB
BB10 34 21 1D BB 34 10 EE FF 28 35 30 2C 35 30 29 2C
BB20 36 00
h]■
```

【図7・20】――ダンプ・リスト

マシン語を入力し、

GBB00 ⏎

で走らせてください。次のような実行結果が得られるでしょう。

【図7・21】――実行終了

ここで強調したいことは、システム・サブルーチンの利用により、マシン語でもきわめて簡単に

カラー・グラフィックが扱える!

ということです。

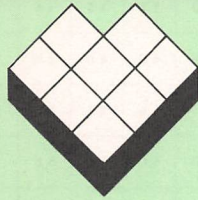
これで、**「PC-8801マシン語活用マニュアル」**

じ・えんど!

です。皆さんのご健闘をお祈りして、めでたく

バイチャ!

第8章

ふろく

Z-80活用表——1

(●本表は「MCOM-82インストラクション活用表(日本電気KK)により作成しました)

8ビット

×	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	n	(nn)	I	R
LD A,×	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3E n	3A n n	ED 57	ED 5F
LD B,×	47	40	41	42	43	44	45	46			DD 46 d	FD 46 d	06 n			
LD C,×	4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d	0E n			
LD D,×	57	50	51	52	53	54	55	56			DD 56 d	FD 56 d	16 n			
LD E,×	5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d	1E n			
LD H,×	67	60	61	62	63	64	65	66			DD 66 d	FD 66 d	26 n			
LD L,×	6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d	2E n			
LD (HL),×	77	70	71	72	73	74	75						36 n			
LD (BC),×	02															
LD (DE),×	12															
LD (IX+d),×	DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d						DD 36 d n			
LD (IY+d),×	FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d						FD 36 d n			
LD (nn),×	32 n n															
LD I,×	ED 47															
LD R,×	ED 4F															
ADD A,×	87	80	81	82	83	84	85	86			DD 86 d	FD 86 d	C6 n			
ADC A,×	8F	88	89	8A	8B	8C	8D	8E			DD 8E d	FD 8E d	CE n			
SUB ×	97	90	91	92	93	94	95	96			DD 96 d	FD 96 d	D6 n			
SBC A,×	9F	98	99	9A	9B	9C	9D	9E			DD 9E d	FD 9E d	DE n			
AND ×	A7	A0	A1	A2	A3	A4	A5	A6			DD A6 d	FD A6 d	E6 n			
XOR ×	AF	A8	A9	AA	AB	AC	AD	AE			DD AE d	FD AE d	EE n			
OR ×	B7	B0	B1	B2	B3	B4	B5	B6			DD B6 d	FD B6 d	F6 n			
CP ×	BF	B8	B9	BA	BB	BC	BD	BE			DD BE d	FD BE d	FE n			
INC ×	3C	04	0C	14	1C	24	2C	34			DD 34 d	FD 34 d				
DEC ×	3D	05	0D	15	1D	25	2D	35			DD 35 d	FD 35 d				

Z−80活用表——2

回転

×	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
RLC ×	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
RRC ×	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
RL ×	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
RR ×	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
SLA ×	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
SRA ×	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
SRL ×	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
RLD								ED 6F		
RRD								ED 67		

RLCA	07
RRCA	0F
RLA	17
RRA	1F

16ビット

×	BC	DE	HL	SP	IX	IY	AF	nn	(nn)
LD AF, ×									
LD BC, ×								01 n n	ED 4B n n
LD DE, ×								11 n n	ED 5B n n
LD HL, ×								21 n n	2A n n
LD SP, ×			F9		DD F9	FD F9		31 n n	ED 7B n n
LD IX, ×								DD 21 n n	DD 2A n n
LD IY, ×								FD 21 n n	FD 2A n n
LD (nn), ×	ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH ×	C5	D5	E5		DD E5	FD E5	F5		
POP ×	C1	D1	E1		DD E1	FD E1	F1		
ADD HL, ×	09	19	29	39					
ADD IX, ×	DD 09	DD 19		DD 39	DD 29				
ADD IY, ×	FD 09	FD 19		FD 39		FD 29			
ADC HL, ×	ED 4A	ED 5A	ED 6A	ED 7A					
SBC HL, ×	ED 42	ED 52	ED 62	ED 72					
INC ×	03	13	23	33	DD 23	FD 23			
DEC ×	0B	1B	2B	3B	DD 2B	FD 2B			

Z−80活用表——3

ジャンプ，コール，リターン

×	UN COND	C	NC	Z	NZ	PE	PO	M	P	
JP ×, nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JR ×, e	18 e−2	38 e−2	30 e−2	28 e−2	20 e−2					
JP (HL)	E9									
JP (IX)	DD E9									
JP (IY)	FD E9									
CALL ×, nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DJNZ e										10 e−2
RET ×	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETI	ED 4D									
RETN	ED 45									

入力

IN A, n	DB n
IN A, (C)	ED 78
IN B, (C)	ED 40
IN C, (C)	ED 48
IN D, (C)	ED 50
IN E, (C)	ED 58
IN H, (C)	ED 60
IN L, (C)	ED 68
INI	ED A2
INIR	ED B2
IND	ED AA
INDR	ED BA

ブロック・サーチ

CPI	ED A1
CPIR	ED B1
CPD	ED A9
CPDR	ED B9

ブロック転送

LDI	ED A0
LDIR	ED B0
LDD	ED A8
LDDR	ED B8

アキュムレータ操作

DAA	27
CPL	2F
NEG	ED 44
CCF	3F
SCF	37

エクスチェンジ

EX AF, AF'	08
EX DE, HL	EB
EX (SP), HL	E3
EX (SP), IX	DD E3
EX (SP), IY	FD E3
EXX	D9

リスタート

RST 00H	C7
RST 08H	CF
RST 10H	D7
RST 18H	DF
RST 20H	E7
RST 28H	EF
RST 30H	F7
RST 38H	FF

CPUコントロール

NOP	00
HALT	76
DI	F3
EI	FB
IM 0	ED 46
IM 1	ED 56
IM 2	ED 5E

出力

OUT n, A	D3 n
OUT (C), A	ED 79
OUT (C), B	ED 41
OUT (C), C	ED 49
OUT (C), D	ED 51
OUT (C), E	ED 59
OUT (C), H	ED 61
OUT (C), L	ED 69
OUTI	ED A3
OTIR	ED B3
OUTD	ED AB
OTDR	ED BB

Z−80活用法

ビット操作

×	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
BIT 0, ×	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
BIT 1, ×	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
BIT 2, ×	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
BIT 3, ×	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
BIT 4, ×	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
BIT 5, ×	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
BIT 6, ×	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
BIT 7, ×	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RES 0, ×	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
RES 1, ×	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
RES 2, ×	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
RES 3, ×	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
RES 4, ×	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
RES 5, ×	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
RES 6, ×	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
RES 7, ×	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET 0, ×	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
SET 1, ×	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
SET 2, ×	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
SET 3, ×	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
SET 4, ×	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
SET 5, ×	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
SET 6, ×	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
SET 7, ×	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

マシン語←→ニーモニック対応表—1

機械語←→ニーモニック											
00	NOP		40	LD	B,B	80	ADD	A,B	C0	RET	NZ
01	LD	BC,nn	41	LD	B,C	81	ADD	A,C	C1	POP	BC
02	LD	(BC),A	42	LD	B,D	82	ADD	A,D	C2	JP	NZ,nn
03	INC	BC	43	LD	B,E	83	ADD	A,E	C3	JP	nn
04	INC	B	44	LD	B,H	84	ADD	A,H	C4	CALL	NZ,nn
05	DEC	B	45	LD	B,L	85	ADD	A,L	C5	PUSH	BC
06	LD	B,n	46	LD	B,(HL)	86	ADD	A,(HL)	C6	ADD	A,n
07	RLCA		47	LD	B,A	87	ADD	A,A	C7	RST	00H
08	EX	AF,AF'	48	LD	C,B	88	ADC	A,B	C8	RET	Z
09	ADD	HL,BC	49	LD	C,C	89	ADC	A,C	C9	RET	
0A	LD	A,(BC)	4A	LD	C,D	8A	ADC	A,D	CA	JP	Z,nn
0B	DEC	BC	4B	LD	C,E	8B	ADC	A,E	CB		
0C	INC	C	4C	LD	C,H	8C	ADC	A,H	CC	CALL	Z,nn
0D	DEC	C	4D	LD	C,L	8D	ADC	A,L	CD	CALL	nn
0E	LD	C,n	4E	LD	C,(HL)	8E	ADC	A,(HL)	CE	ADC	A,n
0F	RRCA		4F	LD	C,A	8F	ADC	A,A	CF	RST	08H
10	DJNZ	e	50	LD	D,B	90	SUB	B	D0	RET	NC
11	LD	DE,nn	51	LD	D,C	91	SUB	C	D1	POP	DE
12	LD	(DE),A	52	LD	D,D	92	SUB	D	D2	JP	NC,nn
13	INC	DE	53	LD	D,E	93	SUB	E	D3	OUT	n,A
14	INC	D	54	LD	D,H	94	SUB	H	D4	CALL	NC,nn
15	DEC	D	55	LD	D,L	95	SUB	L	D5	PUSH	DE
16	LD	D,n	56	LD	D,(HL)	96	SUB	(HL)	D6	SUB	n
17	RLA		57	LD	D,A	97	SUB	A	D7	RST	10H
18	JR	e	58	LD	E,B	98	SBC	A,B	D8	RET	C
19	ADD	HL,DE	59	LD	E,C	99	SBC	A,C	D9	EXX	
1A	LD	A,(DE)	5A	LD	E,D	9A	SBC	A,D	DA	JP	C,nn
1B	DEC	DE	5B	LD	E,E	9B	SBC	A,E	DB	IN	A,n
1C	INC	E	5C	LD	E,H	9C	SBC	A,H	DC	CALL	C,nn
1D	DEC	E	5D	LD	E,L	9D	SBC	A,L	DD		
1E	LD	E,n	5E	LD	E,(HL)	9E	SBC	A,(HL)	DE	SBC	A,n
1F	RRA		5F	LD	E,A	9F	SBC	A,A	DF	RST	18H
20	JR	NZ,e	60	LD	H,B	A0	AND	B	E0	RET	PO
21	LD	HL,nn	61	LD	H,C	A1	AND	C	E1	POP	HL
22	LD	(nn),HL	62	LD	H,D	A2	AND	D	E2	JP	PO,nn
23	INC	HL	63	LD	H,E	A3	AND	E	E3	EX	(SP),HL
24	INC	H	64	LD	H,H	A4	AND	H	E4	CALL	PO,nn
25	DEC	H	65	LD	H,L	A5	AND	L	E5	PUSH	HL
26	LD	H,n	66	LD	H,(HL)	A6	AND	(HL)	E6	AND	n
27	DAA		67	LD	H,A	A7	AND	A	E7	RST	20H
28	JR	Z,e	68	LD	L,B	A8	XOR	B	E8	RET	PE
29	ADD	HL,HL	69	LD	L,C	A9	XOR	C	E9	JP	(HL)
2A	LD	HL,(nn)	6A	LD	L,D	AA	XOR	D	EA	JP	PE,nn
2B	DEC	HL	6B	LD	L,E	AB	XOR	E	EB	EX	DE,HL
2C	INC	L	6C	LD	L,H	AC	XOR	H	EC	CALL	PE,nn
2D	DEC	L	6D	LD	L,L	AD	XOR	L	ED		
2E	LD	L,n	6E	LD	L,(HL)	AE	XOR	(HL)	EE	XOR	n
2F	CPL		6F	LD	L,A	AF	XOR	A	EF	RST	28H
30	JR	NC,e	70	LD	(HL),B	B0	OR	B	F0	RET	P
31	LD	SP,nn	71	LD	(HL),C	B1	OR	C	F1	POP	AF
32	LD	(nn),A	72	LD	(HL),D	B2	OR	D	F2	JP	P,nn
33	INC	SP	73	LD	(HL),E	B3	OR	E	F3	DI	
34	INC	(HL)	74	LD	(HL),H	B4	OR	H	F4	CALL	P,nn
35	DEC	(HL)	75	LD	(HL),L	B5	OR	L	F5	PUSH	AF
36	LD	(HL),n	76	HALT		B6	OR	(HL)	F6	OR	n
37	SCF		77	LD	(HL),A	B7	OR	A	F7	RST	30H
38	JR	C,e	78	LD	A,B	B8	CP	B	F8	RET	M
39	ADD	HL,SP	79	LD	A,C	B9	CP	C	F9	LD	SP,HL
3A	LD	A,(nn)	7A	LD	A,D	BA	CP	D	FA	JP	M,nn
3B	DEC	SP	7B	LD	A,E	BB	CP	E	FB	EI	
3C	INC	A	7C	LD	A,H	BC	CP	H	FC	CALL	M,nn
3D	DEC	A	7D	LD	A,L	BD	CP	L	FD		
3E	LD	A,n	7E	LD	A,(HL)	BE	CP	(HL)	FE	CP	n
3F	CCF		7F	LD	A,A	BF	CP	A	FF	RST	38H

マシン語←→ニーモニック対応表——2

CB ××			
00 RLC B	40 BIT 0,B	80 RES 0,B	C0 SET 0,B
01 RLC C	41 BIT 0,C	81 RES 0,C	C1 SET 0,C
02 RLC D	42 BIT 0,D	82 RES 0,D	C2 SET 0,D
03 RLC E	43 BIT 0,E	83 RES 0,E	C3 SET 0,E
04 RLC H	44 BIT 0,H	84 RES 0,H	C4 SET 0,H
05 RLC L	45 BIT 0,L	85 RES 0,L	C5 SET 0,L
06 RLC (HL)	46 BIT 0,(HL)	86 RES 0,(HL)	C6 SET 0,(HL)
07 RLC A	47 BIT 0,A	87 RES 0,A	C7 SET 0,A
08 RRC B	48 BIT 1,B	88 RES 1,B	C8 SET 1,B
09 RRC C	49 BIT 1,C	89 RES 1,C	C9 SET 1,C
0A RRC D	4A BIT 1,D	8A RES 1,D	CA SET 1,D
0B RRC E	4B BIT 1,E	8B RES 1,E	CB SET 1,E
0C RRC H	4C BIT 1,H	8C RES 1,H	CC SET 1,H
0D RRC L	4D BIT 1,L	8D RES 1,L	CD SET 1,L
0E RRC (HL)	4E BIT 1,(HL)	8E RES 1,(HL)	CE SET 1,(HL)
0F RRC A	4F BIT 1,A	8F RES 1,A	CF SET 1,A
10 RL B	50 BIT 2,B	90 RES 2,B	D0 SET 2,B
11 RL C	51 BIT 2,C	91 RES 2,C	D1 SET 2,C
12 RL D	52 BIT 2,D	92 RES 2,D	D2 SET 2,D
13 RL E	53 BIT 2,E	93 RES 2,E	D3 SET 2,E
14 RL H	54 BIT 2,H	94 RES 2,H	D4 SET 2,H
15 RL L	55 BIT 2,L	95 RES 2,L	D5 SET 2,L
16 RL (HL)	56 BIT 2,(HL)	96 RES 2,(HL)	D6 SET 2,(HL)
17 RL A	57 BIT 2,A	97 RES 2,A	D7 SET 2,A
18 RR B	58 BIT 3,B	98 RES 3,B	D8 SET 3,B
19 RR C	59 BIT 3,C	99 RES 3,C	D9 SET 3,C
1A RR D	5A BIT 3,D	9A RES 3,D	DA SET 3,D
1B RR E	5B BIT 3,E	9B RES 3,E	DB SET 3,E
1C RR H	5C BIT 3,H	9C RES 3,H	DC SET 3,H
1D RR L	5D BIT 3,L	9D RES 3,L	DD SET 3,L
1E RR (HL)	5E BIT 3,(HL)	9E RES 3,(HL)	DE SET 3,(HL)
1F RR A	5F BIT 3,A	9F RES 3,A	DF SET 3,A
20 SLA B	60 BIT 4,B	A0 RES 4,B	E0 SET 4,B
21 SLA C	61 BIT 4,C	A1 RES 4,C	E1 SET 4,C
22 SLA D	62 BIT 4,D	A2 RES 4,D	E2 SET 4,D
23 SLA E	63 BIT 4,E	A3 RES 4,E	E3 SET 4,E
24 SLA H	64 BIT 4,H	A4 RES 4,H	E4 SET 4,H
25 SLA L	65 BIT 4,L	A5 RES 4,L	E5 SET 4,L
26 SLA (HL)	66 BIT 4,(HL)	A6 RES 4,(HL)	E6 SET 4,(HL)
27 SLA A	67 BIT 4,A	A7 RES 4,A	E7 SET 4,A
28 SRA B	68 BIT 5,B	A8 RES 5,B	E8 SET 5,B
29 SRA C	69 BIT 5,C	A9 RES 5,C	E9 SET 5,C
2A SRA D	6A BIT 5,D	AA RES 5,D	EA SET 5,D
2B SRA E	6B BIT 5,E	AB RES 5,E	EB SET 5,E
2C SRA H	6C BIT 5,H	AC RES 5,H	EC SET 5,H
2D SRA L	6D BIT 5,L	AD RES 5,L	ED SET 5,L
2E SRA (HL)	6E BIT 5,(HL)	AE RES 5,(HL)	EE SET 5,(HL)
2F SRA A	6F BIT 5,A	AF RES 5,A	EF SET 5,A
30	70 BIT 6,B	B0 RES 6,B	F0 SET 6,B
31	71 BIT 6,C	B1 RES 6,C	F1 SET 6,C
32	72 BIT 6,D	B2 RES 6,D	F2 SET 6,D
33	73 BIT 6,E	B3 RES 6,E	F3 SET 6,E
34	74 BIT 6,H	B4 RES 6,H	F4 SET 6,H
35	75 BIT 6,L	B5 RES 6,L	F5 SET 6,L
36	76 BIT 6,(HL)	B6 RES 6,(HL)	F6 SET 6,(HL)
37	77 BIT 6,A	B7 RES 6,A	F7 SET 6,A
38 SRL B	78 BIT 7,B	B8 RES 7,B	F8 SET 7,B
39 SRL C	79 BIT 7,C	B9 RES 7,C	F9 SET 7,C
3A SRL D	7A BIT 7,D	BA RES 7,D	FA SET 7,D
3B SRL E	7B BIT 7,E	BB RES 7,E	FB SET 7,E
3C SRL H	7C BIT 7,H	BC RES 7,H	FC SET 7,H
3D SRL L	7D BIT 7,L	BD RES 7,L	FD SET 7,L
3E SRL (HL)	7E BIT 7,(HL)	BE RES 7,(HL)	FE SET 7,(HL)
3F SRL A	7F BIT 7,A	BF RES 7,A	FF SET 7,A

マシン語←→ニーモニック対応表——3

DD XX		
09	ADD	IX, BC
19	ADD	IX, DE
21	LD	IX, nn
22	LD	(nn), IX
23	INC	IX
29	ADD	IX, IX
2A	LD	IX, (nn)
2B	DEC	IX
34	INC	(IX+d)
35	DEC	(IX+d)
36	LD	(IX+d), n
39	ADD	IX, SP
46	LD	B, (IX+d)
4E	LD	C, (IX+d)
56	LD	D, (IX+d)
5E	LD	E, (IX+d)
66	LD	H, (IX+d)
6E	LD	L, (IX+d)
70	LD	(IX+d), B
71	LD	(IX+d), C
72	LD	(IX+d), D
73	LD	(IX+d), E
74	LD	(IX+d), H
75	LD	(IX+d), L
77	LD	(IX+d), A
7E	LD	A, (IX+d)
86	ADD	A, (IX+d)
8E	ADC	A, (IX+d)
96	SUB	(IX+d)
9E	SBC	A, (IX+d)
A6	AND	(IX+d)
AE	XOR	(IX+d)
B6	OR	(IX+d)
BE	CP	(IX+d)
CB d 06	RLC	(IX+d)
CB d 0E	RRC	(IX+d)
CB d 16	RL	(IX+d)
CB d 1E	RR	(IX+d)
CB d 26	SLA	(IX+d)
CB d 2E	SRA	(IX+d)
CB d 3E	SRL	(IX+d)
CB d 46	BIT	0, (IX+d)
CB d 4E	BIT	1, (IX+d)
CB d 56	BIT	2, (IX+d)
CB d 5E	BIT	3, (IX+d)
CB d 66	BIT	4, (IX+d)
CB d 6E	BIT	5, (IX+d)
CB d 76	BIT	6, (IX+d)
CB d 7E	BIT	7, (IX+d)
CB d 86	RES	0, (IX+d)
CB d 8E	RES	1, (IX+d)
CB d 96	RES	2, (IX+d)
CB d 9E	RES	3, (IX+d)
CB d A6	RES	4, (IX+d)
CB d AE	RES	5, (IX+d)
CB d B6	RES	6, (IX+d)
CB d BE	RES	7, (IX+d)
CB d C6	SET	0, (IX+d)
CB d CE	SET	1, (IX+d)
CB d D6	SET	2, (IX+d)
CB d DE	SET	3, (IX+d)
CB d E6	SET	4, (IX+d)
CB d EE	SET	5, (IX+d)
CB d F6	SET	6, (IX+d)
CB d FE	SET	7, (IX+d)
E1	POP	IX
E3	EX	(SP), IX
E5	PUSH	IX
E9	JP	(IX)
F9	LD	SP, IX

ED XX		
40	IN	B, (C)
41	OUT	(C), B
42	SBC	HL, BC
43	LD	(nn), BC
44	NEG	
45	RETN	
46	IM	0
47	LD	I, A
48	IN	C, (C)
49	OUT	(C), C
4A	ADC	HL, BC
4B	LD	BC, (nn)
4D	RETI	
4F	LD	R, A
50	IN	D, (C)
51	OUT	(C), D
52	SBC	HL, DE
53	LD	(nn), DE
56	IM	1
57	LD	A, I
58	IN	E, (C)
59	OUT	(C), E
5A	ADC	HL, DE
5B	LD	DE, (nn)
5E	IM	2
5F	LD	A, R
60	IN	H, (C)
61	OUT	(C), H
62	SBC	HL, HL
67	RRD	
68	IN	L, (C)
69	OUT	(C), L
6A	ADC	HL, HL
6F	RLD	
72	SBC	HL, SP
73	LD	(nn), SP
78	IN	A, (C)
79	OUT	(C), A
7A	ADC	HL, SP
7B	LD	SP, (nn)
A0	LDI	
A1	CPI	
A2	INI	
A3	OUTI	
A8	LDD	
A9	CPD	
AA	IND	
AB	OUTD	
B0	LDIR	
B1	CPIR	
B2	INIR	
B3	OTIR	
B8	LDDR	
B9	CPDR	
BA	INDR	
BB	OTDR	

FD XX		
09	ADD	IY, BC
19	ADD	IY, DE
21	LD	IY, nn
22	LD	(nn), IY
23	INC	IY
29	ADD	IY, IY
2A	LD	IY, (nn)
2B	DEC	IY
34	INC	(IY+d)
35	DEC	(IY+d)
36	LD	(IY+d), n
39	ADD	IY, SP
46	LD	B, (IY+d)
4E	LD	C, (IY+d)
56	LD	D, (IY+d)
5E	LD	E, (IY+d)
66	LD	H, (IY+d)
6E	LD	L, (IY+d)
70	LD	(IY+d), B
71	LD	(IY+d), C
72	LD	(IY+d), D
73	LD	(IY+d), E
74	LD	(IY+d), H
75	LD	(IY+d), L
77	LD	(IY+d), A
7E	LD	A, (IY+d)
86	ADD	A, (IY+d)
8E	ADC	A, (IY+d)
96	SUB	(IY+d)
9E	SBC	A, (IY+d)
A6	AND	(IY+d)
AE	XOR	(IY+d)
B6	OR	(IY+d)
BE	CP	(IY+d)
CB d 06	RLC	(IY+d)
CB d 0E	RRC	(IY+d)
CB d 16	RL	(IY+d)
CB d 1E	RR	(IY+d)
CB d 26	SLA	(IY+d)
CB d 2E	SRA	(IY+d)
CB d 3E	SRL	(IY+d)
CB d 46	BIT	0, (IY+d)
CB d 4E	BIT	1, (IY+d)
CB d 56	BIT	2, (IY+d)
CB d 5E	BIT	3, (IY+d)
CB d 66	BIT	4, (IY+d)
CB d 6E	BIT	5, (IY+d)
CB d 76	BIT	6, (IY+d)
CB d 7E	BIT	7, (IY+d)
CB d 86	RES	0, (IY+d)
CB d 8E	RES	1, (IY+d)
CB d 96	RES	2, (IY+d)
CB d 9E	RES	3, (IY+d)
CB d A6	RES	4, (IY+d)
CB d AE	RES	5, (IY+d)
CB d B6	RES	6, (IY+d)
CB d BE	RES	7, (IY+d)
CB d C6	SET	0, (IY+d)
CB d CE	SET	1, (IY+d)
CB d D6	SET	2, (IY+d)
CB d DE	SET	3, (IY+d)
CB d E6	SET	4, (IY+d)
CB d EE	SET	5, (IY+d)
CB d F6	SET	6, (IY+d)
CB d FE	SET	7, (IY+d)
E1	POP	IY
E3	EX	(SP), IY
E5	PUSH	IY
E9	JP	(IY)
F9	LD	SP, IY

あとがき

本書は、ＰＣ－８８０１を使ってマシン語を活用しようとするユーザーに対し、マシン語プログラマーへの技術情報を提供しようとするものです。各周辺機器の制御にあたっては、ユーザーの便宜を考え、〈システム・サブルーチン〉を可能な限り活用する方法をとりました。このため、直接Ｉ／Ｏポートを制御する方法は省略いたしました。また、〈漢字の制御〉など、紙面の都合上割愛した部分もあります。これらの資料については、また別の機会にまとめたいと思います。

本書は、その性格上、マシン語の知識を前提としています。そこでマシン語未修得者のために、以下に参考文献を掲げておきます。また、ＰＣ－８８０１に関する資料など、合わせて紹介しておきます。

マシン語の入門書

❶ 『ＰＣ－８００１マシン語入門Ｉ、II』
(塚越一雄著、電波新聞社)

ＰＣ－８８０１に関する情報

❷ 『ＰＣ－８８０１ N88－ＢＡＳＩＣ ＡＮＡＬＹＴＩＣＡＬ ＭＡＮＵＡＬ』
(川村清著、秀和システムトレーデイング)

❸ 『ＰＣ－Techknow ８８００ Vol 1.1』(アスキー)

Ｚ－80関係資料

❹ 『μＣＯＭ－82ユーザーズ・マニュアル』(日本電気)

❺ 『μＣＯＭ－82インストラクション活用表』(日本電気)

1983年７月14日
ＭＵＬＴＩマイコン研究会事務局代表
塚越一雄

PC-8801マシン語活用マニュアル

著 者 塚越一雄 © Kazuo Tsukagoshi 1983
発行者 田村正隆

発行所 株式会社ナツメ社
東京都千代田区神田神保町1－52(〒101)
電話 03(291)1257(代表)
振替 東京3-58661
印 刷 ラン印刷社
製 本 文章堂製本

ISBN4-8163-0326-X